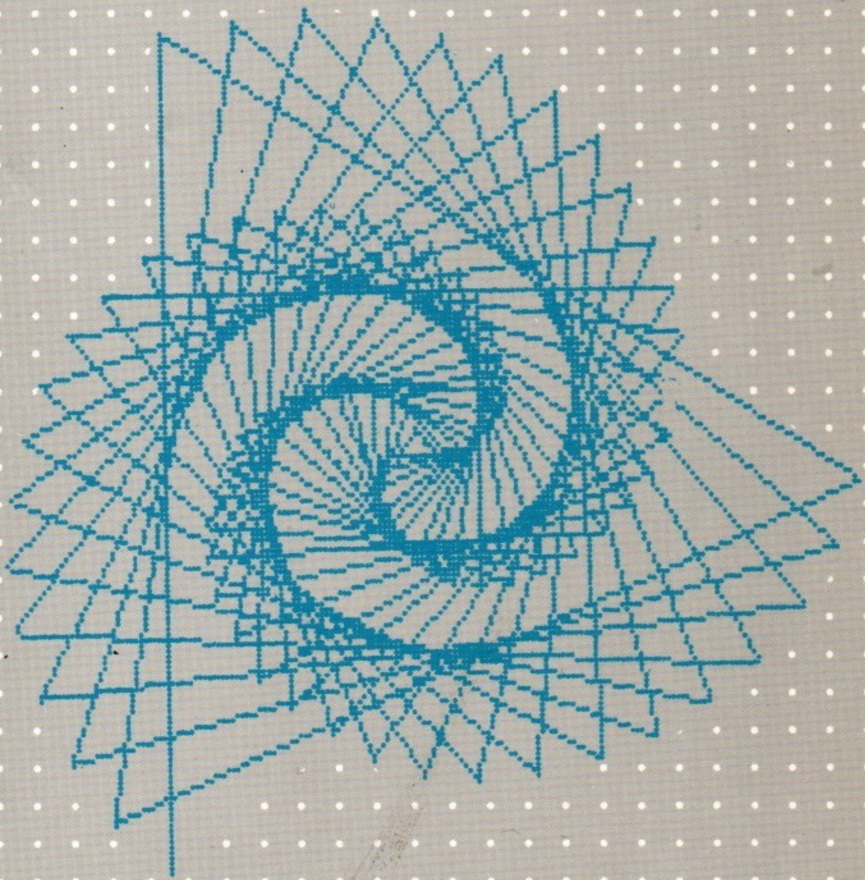
Marcel Sutter

40 grafische programma's in MSX BASIC

# 40 grafische programma's in MSX BASIC

Leer programmeren
met hoge resolutie graphics
in MSX BASIC

Marcel Sutter



ACADEMIC SERVICE

40 grafische programma's
in MSX BASIC

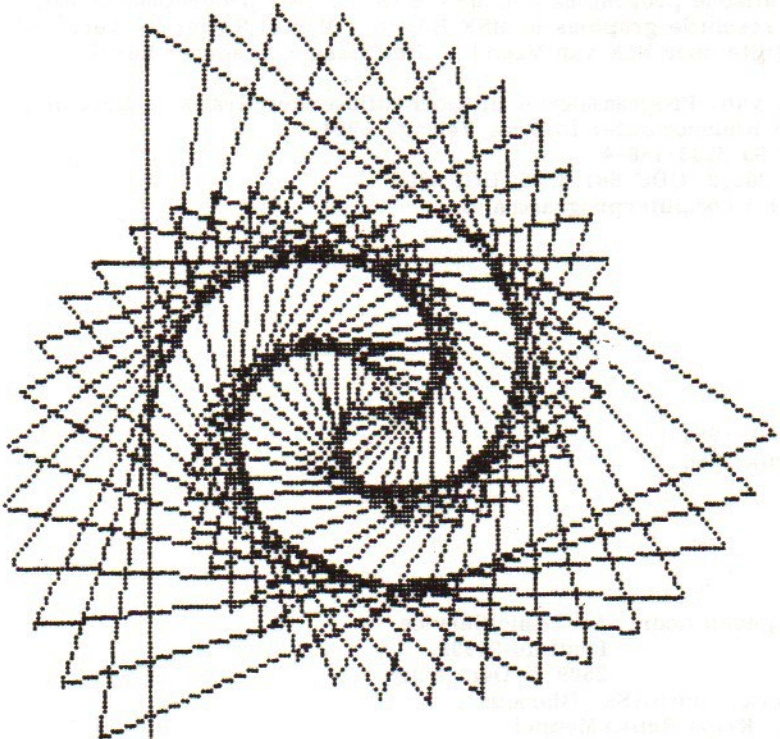
In deze reeks zijn verschenen:

40 grafische programma's voor de Commodore 64
40 grafische programma's in MSX BASIC
40 grafische programma's voor de Apple II, IIe, IIc
40 grafische programma's voor de Electron en BBC
40 grafische programma's in IBM- en GW-BASIC
40 grafische programma's voor de ZX Spectrum

# 40 grafische programma's in MSX BASIC

Leer programmeren
met hoge resolutie graphics
in MSX BASIC

Marcel Sutter



ACADEMIC SERVICE

Oorspronkelijke titel: *Programmieren mit hochauflösender Grafik*
Verschenen bij: Mikro+Kleincomputer Informa Verlag AG, Luzern


Vertaling en bewerking: Nok van Veen

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Sutter, Marcel

40 grafische programma's in MSX BASIC : leer programmeren met hoge resolutie graphics in MSX BASIC / Marcel Sutter ; [vert. uit het Duits door Nok van Veen]. - Den Haag : Academic Service. - Ill.
Vert. van: Programmieren mit hochauflösender Grafik. - Luzern : Mikro-Kleincomputer Informa Verlag, 1984.
ISBN 90-6233-156-4
SISO 365.2 UDC 681.3.06 UGI 650
Trefw.: computerprogramma's.

1e druk 1985
2e druk 1986

Uitgegeven door: Academic Service
Postbus 96996
2509 JJ Den Haag
Zetwerk: multitASk, Blaricum
Druk: Krips Repro Meppel
Bindwerk: Meeuwis, Amsterdam
Omslagontwerp: Leo Bolt

# *Voorwoord*

Door het toenemende gebruik van microcomputers maken de grafische toepassingen een sterke ontwikkeling door. Het beeldscherm is tegenwoordig, veel meer dan de printer, het afdrukapparaat bij uitstek. Grafische toepassingen (graphics) vinden we bij computerspelletjes, bij computerkunst en in toenemende mate op de werkplek van ingenieurs, ontwerpers, architecten en anderen die zich bezighouden met Computer Ondersteund Ontwerpen (Computer Aided Design).

De geïnteresseerde leek is vaak verrukt van de mooie 'plaatjes', die bij demonstraties getoond worden, en denkt dat hiervoor hele ingewikkelde programma's nodig zijn. Iemand, die iets van wiskunde afweet, weet hoe een functie eruitziet en kan doorgaans zonder veel moeite grafische programma's maken.

Ik heb voor het Zwitserse Computertijdschrift 'MIKRO+KLEINCOMPUTER' een cursus 'programmeren met hoge resolutie' geschreven, die in een aantal afleveringen van het blad is verschenen. Al direct na het verschijnen van het eerste artikel werd ik werkelijk overstelpt met enthousiaste reacties van de lezers. Dit heeft ons aangezet tot het maken van dit boek.

Veel van de grafische toepassingsprogramma's worden geschreven voor een bepaald merk en type microcomputer of voor een bepaalde plotter. Wie een dergelijk programma wil herschrijven voor een ander merk microcomputer zal veel moeilijkheden ondervinden en vaak zelfs zijn pogingen staken.

De veertig programma's in dit boek zijn geschreven in standaard BASIC en gebruiken geen POKE- en PEEK-opdrachten. Bovendien worden slechts twee grafische opdrachten gebruikt, te weten het inschakelen van de hoge-resolutiestand en het verbinden van twee punten door een rechte lijn. De illustraties die als voorbeeld dienen van hetgeen de programma's tekenen zijn gemaakt op de miniplotter van de Sharp PC-1500.

Wij hopen dat de lezers veel plezier aan deze programma's zullen beleven en dat zij voor velen een aanmoediging zullen zijn om het terrein van de computergraphics verder te verkennen. Mijn dank gaat uit naar de heer H.J. Ottenbacher van Micro+Kleincomputer; zonder hem zou dit boek nooit het daglicht hebben gezien.

Marcel Sutter
voorjaar 1984

# *Voorwoord bij de Nederlandse uitgave*

De auteur, Marcel Sutter, heeft dit boek oorspronkelijk geschreven in standaard Microsoft-BASIC. In de programma's maakte hij alleen gebruik van de opdracht om twee punten met elkaar te verbinden. In de MSX-versie die wij van zijn boek hebben gemaakt geven we daarentegen suggesties voor het werken met kleuren en laten we bij een groot aantal programma's zien hoe u met de CIRCLE- en PAINT-opdrachten de programma's kunt uitbreiden en verfraaien. In de appendix vindt u een aantal uitgewerkte voorbeelden. Hierin laten we ook zien hoe u tekst en getallen op het grafische scherm kunt afdrukken.

Alle programma's zijn ingetoetst op een MSX-computer en direct vanuit het geheugen van de computer op een printer afgedrukt. Als u de programma's precies zo intoetst als ze in het boek staan afgedrukt, moeten ze allemaal goed werken. De programma's zijn ingetoetst met het toetsenbord in de 'kleine-letterstand'. Hierdoor wordt de tekst in de commentaarregels en in INPUT- en PRINT-opdrachten in kleine letters afgedrukt, terwijl de computer van alle BASIC-opdrachten (zoals rem, print, for, input, line, enz.) en van alle variabelen (zoals x1, u, h, k, yy, enz.), die we met kleine letters intoetsen, zelf hoofdletters maakt. U kunt, als u dat wilt natuurlijk, ook alles in de 'hoofdletterstand' intoetsen.

Wij raden u aan reeds bij de eerste programma's te experimenteren met de COLOR-opdracht (verschillende kleuren voor de voorgrond, de achtergrond en de boven- en onderrand), met de LINE-opdracht (verschillende afdrukkleuren), met de PAINT-opdracht (voor het kleuren van vlakken) en met de DRAW-opdracht (voor het afdrukken van tekst op het grafische scherm). U kunt uw ervaringen dan direct bij volgende programma's gebruiken. In de tekst vindt u suggesties hoe u de bovengenoemde opdrachten kunt gebruiken.

De auteur heeft voor de tekeningen gebruik gemaakt van een Sharp-plotter met een resolutie van 220 bij 220 puntjes. Op een aantal plaatsen hebben wij zijn tekeningen vervangen door afdrukken, die

wij zelf van het grafische scherm van de computer gemaakt hebben. Deze afdrukken geven een beter beeld van hoe u de tekeningen en grafieken op uw eigen MSX-scherm ziet.

Alle programma's zijn kort. Verfraai ze, verbeter ze en breid ze uit. De mogelijkheden zijn onbegrensd.

Het is een boek voor elke MSX-computer. Diegenen die de veelal wiskundige uitleg bij de programma's teveel van het goede vinden, zullen alleen al door het intikken en draaien van de programma's verrukt zijn van het resultaat. Anderen, die zich nog iets van de sinus en cosinus uit hun schooltijd herinneren of die nu op school zitten, zullen weinig of geen moeite hebben met de theorie in dit boek. Leraren en leerlingen kunnen wellicht hun voordeel doen met de programma's voor het tekenen van functies. Ook het driedimensionaal tekenen wordt behandeld. Erg leuk zijn de vijf programma's waarmee in BASIC getekend kan worden zoals dat met LOGO ook kan. Dit geeft zeer fraaie grafieken, vooral de turtle-graphics. De laatste vijf programma's zijn toepassingsprogramma's, waarin 'graphics' de hoofdrol spelen.

Hopelijk hebt u net zoveel plezier met het draaien en verfraaien van deze programma's als wij hebben gehad bij de bewerking van dit boek.

Nok van Veen
februari 1985

# Inhoud


| | | |
|---|---|---|
| 1 | MSX-graphics | 1 |
| 2 | Grafieken van functies in cartesische vorm | 15 |
| 3 | Krommen in poolcoördinaten en in parametervorm | 30 |
| 4 | Tekenen van driedimensionale figuren | 49 |
| 5 | Het tekenen van vlakken in de ruimte | 63 |
| 6 | Turtle-graphics en LOGO-simulatie | 82 |
| 7 | Educatieve toepassingsprogramma's | 96 |
| Appendix | | 110 |

programma 4: spiraaleffect (PAINT)
programma 19: echte vliegekop (CIRCLE, PAINT)
programma 21: computerkunst; tekst op het grafische scherm (DRAW)
programma 25: bol met verborgen lijnen en draaiing
programma 36: gekleurde landkaart (COLOR, PAINT)
programma 39: tekst op het grafische scherm (DRAW)

# 1 MSX-graphics

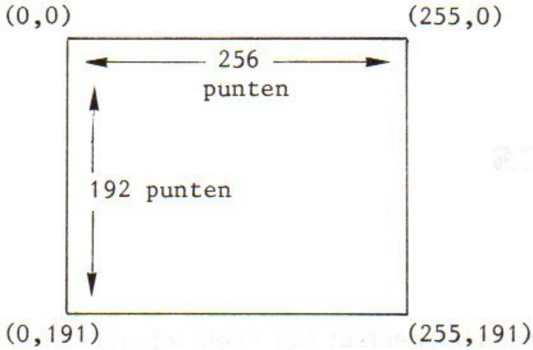
In de Hoge-Resolutiestand (screen 2) bestaat het beeldscherm van een MSX-computer uit 256 beeldpunten horizontaal en 192 beeldpunten verticaal. Vanuit een BASIC-programma kunnen we elk van deze 49152 puntjes aan- en uitzetten (PSET en PRESET); we kunnen lijnen trekken tussen twee beeldpunten (LINE); we kunnen cirkels tekenen (CIRCLE); we kunnen vlakken kleuren (PAINT) en we kunnen van een speciale grafische macrotaal gebruikmaken (DRAW). De hoge-resolutiestand kiezen we met de SCREEN-opdracht, terwijl we de afdrukkleur, de achtergrondkleur en de kaderkleur kunnen instellen met de COLOR-opdracht.

In de programma's gebruiken we in principe de volgende drie grafische opdrachten:

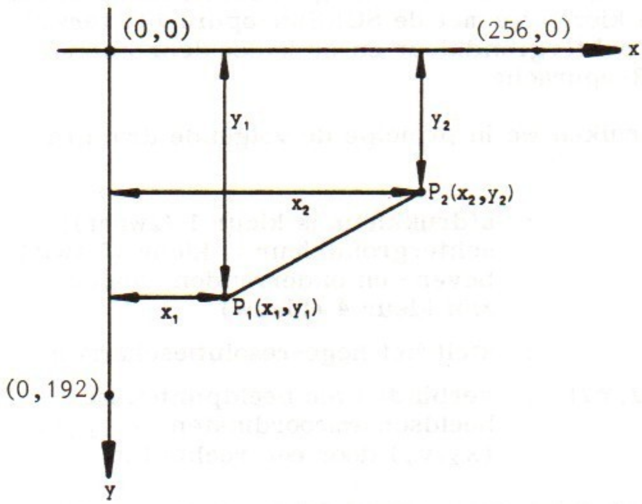
| | | |
|---|---|---|
| COLOR 1,15,4 | : | afdrukkleur is kleur 1 (zwart); achtergrondkleur is kleur 15 (wit); boven- en onderranden (kader) zijn kleur 4 (blauw) |
| SCREEN 2 | : | stelt het hoge-resolutiescherm in |
| LINE (X1,Y1)-(X2,Y2) | : | verbindt twee beeldpunten met beeldschermcoördinaten $(x_1,y_1)$ en $(x_2,y_2)$ door een rechte lijn |

Ook erg mooi is een roodbruine achtergrond (kleur 6) en een gele afdrukkleur (kleur 10 of 11). Dit stellen we in met COLOR 11,6,6. Hierbij is er geen boven- en onderrand maar een achtergrond in één kleur.

Met de SCREEN 2-opdracht krijgen we de beeldschermindeling van p.2 bovenaan. Linksboven in het beeldscherm ligt de oorsprong van het beeldschermcoördinatenstelsel. De bovenrand van het scherm noemen we de X-as. De linkerrand noemen we de Y-as. De Y-as wijst dus naar beneden. Een punt met een grotere Y-coördinaat ligt dus onder (voor ons) een punt met een kleinere Y-coördinaat. Als




middelpunt van het scherm nemen we het punt (U,V) met U=128 en V=96. Wellicht moet u even wennen aan de, naar beneden wijzende, Y-as.




De LINE-opdracht trekt een lijn tussen twee beeldschermpunten in de afdrukkleur die met COLOR is ingesteld. We moeten echter oppassen. Het volgende programmaatje zal op het scherm een rechthoek tekenen en géén vierkant!

```
10 COLOR 1,15,4 : SCREEN 2
20 LINE (50,50)-(150,50)
30 LINE (150,50)-(150,150)
40 LINE (150,150)-(50,150)
50 LINE (50,150)-(50,50)
60 A$=INPUT$(1)
70 END
```

De horizontale zijden zijn 1,4 maal zo lang als de verticale zijden. Als we een vierkant willen tekenen, zullen we alle acht de X-coördinaten in de regels 20-50 door 1,4 moeten delen. Omdat we in vrijwel alle programma's een verhouding van 1 op 1 voor de x- en y-richting willen zien, passen we de X-coördinaten in de LINE-opdrachten (en ook in de CIRCLE- en PAINT-opdrachten) als volgt aan:

$$\text{nieuwe X} = 37 + \frac{\text{oude X}}{1{,}4}$$

Hierdoor komt de rechterbovenhoek (X=255) terecht op het beeldpunt met X = 37 + 255/1,4, dus X=219,143 en de oorsprong (X=0) komt op X=37. Omdat beeldschermcoördinaten altijd gehele getallen zijn, nemen we de INT-functie bij het omrekenen van de X-coördinaten. Regel 20 van het vorige programma zou er dan zo uitzien:

```
20 LINE(INT(37+X1/1.4),Y1)-(INT(37+X2/1.4),Y2)
```

In de programma's doen we het iets anders. We maken van deze omrekening een functie met de DEF FN-opdracht, die er zo uitziet:

```
DEF FNX(X)=INT(37+X/1.4+.5)
```

De X tussen haakjes is het argument van de functie. De functie heet FNX. Dit argument, dat in een programma elke variabele mag zijn, wordt door 1,4 gedeeld; er wordt 37 bij opgeteld en door er nog ½ bij op te tellen voordat de INT-functie genomen wordt, ronden we de nieuwe X-coördinaat af. In het vorige programma zou dit er zo uitzien:

```
10 COLOR 1,15,4 : SCREEN 2
15 DEF FNX(X)=INT(37+X/1.4+.5)
20 LINE (FNX(0),0)-(FNX(100),0)
30 LINE(FNX(100),0)-(FNX(100),100)
   :
   :
   enzovoorts
```

In vrijwel elk programma komt u deze functie tegen. De algemene LINE-opdracht voor het trekken van een lijn tussen de punten $P_1(x_1,y_1)$ en $P_2(x_2,y_2)$ ziet er dus zo uit:

LINE(FNX(X1),Y1)-(FNX(X2),Y2)

Deze LINE-opdracht trekt een lijn in de kleur die met COLOR als afdrukkleur gekozen is. Kijk in uw MSX-handleiding wat u allemaal nog meer met LINE kunt doen.

Bij hoge-resolutie-graphics onderscheiden we **puntgraphics** en **lijn- (of vector-) graphics**.

## Puntgraphics

Bij puntgraphics worden de coördinaten x en y van een bepaald punt berekend en wordt dit punt door een bepaalde graphic-opdracht (PSET(x,y)) op het scherm zichtbaar. Willen we een dergelijk punt weer onzichtbaar maken, dan is daar een andere opdracht (PRESET(x,y)) voor nodig. In dit boek zullen we echter voor het tekenen van figuren niet van deze puntgraphics-techniek gebruik maken.

## Lijngraphics

Bij lijngraphics worden door een bepaalde opdracht, die als machineroutine in de microcomputer aanwezig is, razendsnel twee berekende punten $P_1(X_1,Y_1)$ en $P_2(X_2,Y_2)$ door een rechte lijn met elkaar verbonden. Zo'n rechte verbindingslijn wordt opgebouwd uit een groot aantal kleine horizontale en verticale lijnstukjes. Op een afstand van zo'n 50 cm van het scherm lijken deze verbindingslijnen inderdaad bijna recht.

In onze programma's maken we gebruik van deze lijngraphics. Alles wat we gaan tekenen ontstaat door het steeds verbinden van twee punten door een rechte lijn (of door een recht lijntje). De programma's zijn kort, zodat duidelijk wordt op welke wijze de tekeningen ontstaan. U kunt de programma's zelf uitbreiden en verfraaien. U kunt kleuren gebruiken of regels toevoegen waarmee de door u in te toetsen waarden gecontroleerd worden, zodat dit niet leidt tot het afbreken van het programma. In de appendix vindt u een aantal uitbreidingen van enkele programma's. Hierin ziet u hoe u met kleuren kunt werken, hoe u vlakken vult en hoe u tekst op het grafische scherm kunt afdrukken.

## Programmastructuur

In alle programma's gebruiken we steeds dezelfde variabelen en dezelfde programmastructuur. De verschillende variabelen betekenen:

| | |
|---|---|
| X1,Y1 | coördinaten van het beginpunt |
| X2,Y2 | coördinaten van het eindpunt |
| U,V | oorsprong van het wiskundige coördinatensysteem; dikwijls is dit het midden van het beeldscherm (128,96) |
| H | de waarde 0,5 voor het afronden op helen |
| K | de vermenigvuldigingsfactor voor functiewaarden |
| W,W1 | hoeken bij trigoniometrische functies |
| RD | het getal $\pi/180$ voor het omrekenen van graden in radialen |

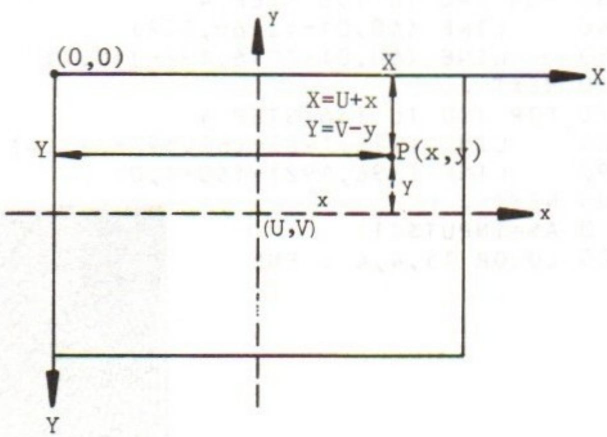
Een grof structuurdiagram voor de programma's ziet er zo uit:

| | |
|---|---|
| begin programma | |
| graphic-stand kiezen | |
| variabelen U,V,H,K,RD,enz. vastleggen | |
| punt P1(X1,Y1) vastleggen | |
| doe zolang nodig | |
| | nieuw punt P2(X2,Y2) berekenen |
| | verbindt P1 met P2 |
| | punt P2 wordt punt P1 (X1=X2 Y1=Y2) |
| einde programma | |

In alle programma's gebruiken we twee belangrijke transformatieformules. Voor het schermcoördinatensysteem van de MSX-computer is de linkerbovenhoek van het scherm de oorsprong. Wijzelf gebruiken echter bijna altijd een coördinatensysteem met de oorsprong in het midden van het scherm. Om de beeldschermcoördinaten X,Y (zoals de computer ze gebruikt) te berekenen uit de coördinaten x,y zoals wij ze gebruiken hanteren we de transformatieformules:

X = INT(U + x + H)   :   Y = INT(V - y + H)

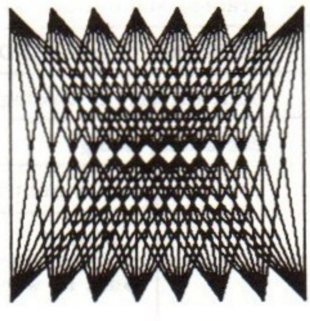
Bekijk de hiernaaststaande figuur eens. Duidelijk is dat elk punt p(x,y) in ons coördinatenstelsel (gestippeld assenkruis) door bovenstaande formules wordt getransformeerd (overgebracht) naar een punt P(X,Y) in het coördinatenstelsel van de computer. Hierbij hoeft onze oorsprong (U,V) natuurlijk niet altijd precies in het midden van het beeld te liggen.




Genoeg theorie, nu de programma's. De ideeën voor de programma's hebben wij gekregen bij het doorbladeren van Amerikaanse, Duitse en Franse computertijdschriften. Alle programma's zijn echter eigen creaties of bewerkingen en vereenvoudigingen van reeds gepubliceerde programma's.

**Programma 1** tekent een 'diagonaalweb'. Elk van de acht bovenste punten wordt door een rechte lijn verbonden met elk van de onderste acht punten.

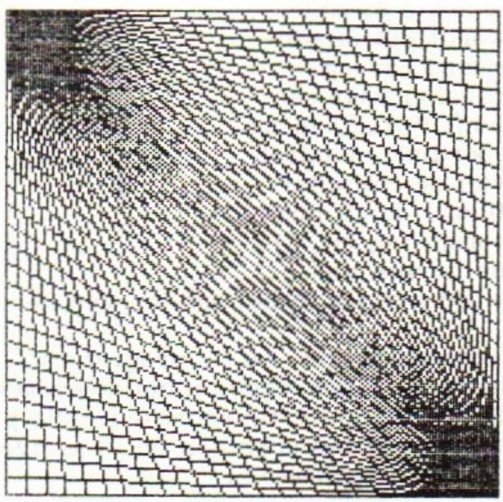
```
100 REM programma 1 diagonaalweb
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 Y1=0 : Y2=192
140 FOR X1= 16 TO 240 STEP 32
150    FOR X2=16 TO 240 STEP 32
160       LINE (X1,Y1)-(X2,Y2)
170    NEXT X2
180 NEXT X1
190 A$=INPUT$(1)
200 COLOR 15,4,4 : END
```



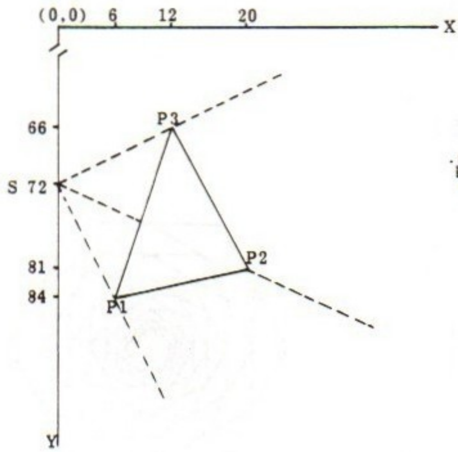
Als u na regel 120 opneemt: `125 DEF FNX(X)=INT(37+X/1.4+.5)`
en regel 160 verandert in: `160 LINE (FNX(X1),Y1)-(FNX(X2),Y2)`
hebben de x- en de y-richting dezelfde orde van grootte.
Het bovenstaande programma geeft een nogal brede figuur.

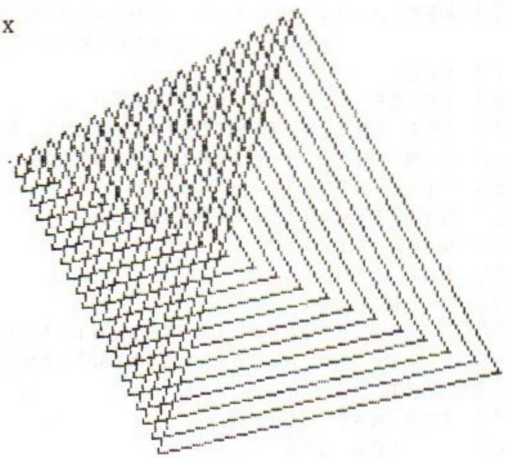
**Programma 2** is een fraai voorbeeld van het moiree-effect. Geniet ervan.

```
100 REM programma 2 moiree-effect
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 FOR J=0 TO 136 STEP 4
140    LINE (60,0)-(J+60,192)
150    LINE (60,0)-(196,192-J*1.4)
160 NEXT J
170 FOR J=0 TO 136 STEP 4
180    LINE (196,192)-(60,192-J*1.4)
190    LINE (196,192)-(60+J,0)
200 NEXT J
210 A$=INPUT$(1)
220 COLOR 15,4,4 : END
```



Programma 3 tekent een reeks driehoeken in perspectief. Het 'centrum van vermenigvuldiging' heeft de coördinaten S(0,72). De drie hoekpunten van de driehoek die vermenigvuldigd wordt zijn P1(6,84), P2(20,81) en P3(12,66). Alle coördinaten zijn beeldschermcoördinaten (oorsprong links bovenaan). De drie richtingen van vermenigvuldiging (richtingsvectoren) zijn $\overrightarrow{SP_1}$, $\overrightarrow{SP_2}$ en $\overrightarrow{SP_3}$. Deze worden in het programma als SX(1);SY(1), SX(2);SY(2) en SX(3);SY(3) vastgelegd. De vermenigvuldigingsfactor K is de lusvariabele van de buitenste FOR-NEXT lus. K loopt van 0 tot 10 in stappen van 0.4. Hieronder zien we de uitgangssituatie. het resultaat van het programma en het programma zelf.



```
100 REM programma 3 driehoeken in perspectief
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DIM X(3),SX(3),SY(3)
140 FOR J=1 TO 3
150     READ SX(J),SY(J)
160 NEXT J
170 X(0)=0 : Y(0)=72
180 FOR K=0 TO 10 STEP .4
190     FOR J=1 TO 3
200         X(J)=X(0)+K*SX(J)
210         Y(J)=Y(0)+K*SY(J)
220     NEXT J
230     LINE (X(1),Y(1))-(X(2),Y(2))
240     LINE (X(2),Y(2))-(X(3),Y(3))
250     LINE (X(3),Y(3))-(X(1),Y(1))
260 NEXT K
270 A$=INPUT$(1)
280 COLOR 15,4,4 : END
290     DATA 6,12,20,9,12,-6
```

Programma 4 tekent een reeks ingeschreven regelmatige zeshoeken. Behalve bij de eerste zeshoek, liggen alle zes de hoeken van elke zeshoek precies in het midden van een zijde van de omgeschreven zeshoek. De coördinaten van de hoekpunten van de eerste (buitenste) zeshoek worden met trigoniometrische functies (sinus- en cosinusfunctie) berekend. De hoek die hierbij als argument van de functies gebruikt wordt doorloopt de waarden 0°, 60°, 120°, 180°, 240°, 300° en 360°. De middelpunten van de zijden van een zeshoek worden aangegeven met MX(0);MY(0) t/m MX(5);MY(5). Dit worden de hoekpunten van de volgende zeshoek.

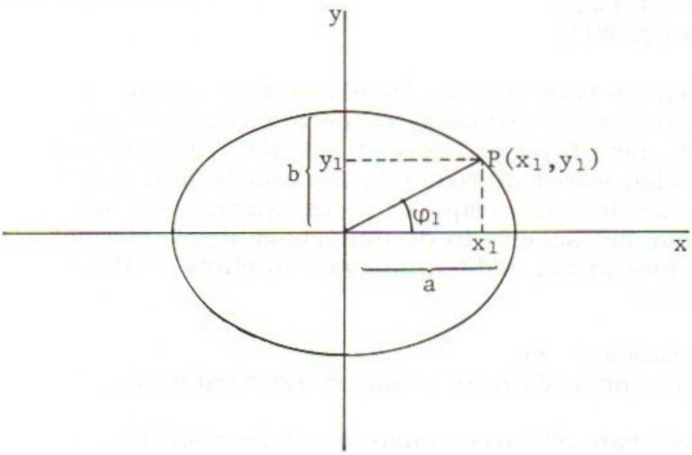
```
100 REM programma 4 ingeschreven
                        zeshoeken
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 DIM X(6),Y(6),MX(6),MY(6)
150 PI=4*ATN(1)
160 U=128:V=96:R=100:H=.5
170 W=60*PI/180
180 FOR J=0 TO 6
190    W1=J*W
200    X(J)=INT(U+R*COS(W1)+H)
210    Y(J)=INT(V-R*SIN(W1)+H)
220 NEXT J
230 FOR N=1 TO 20
240    FOR J=0 TO 5
250      LINE (FNX(X(J)),Y(J))-
              (FNX(X(J+1)),Y(J+1))
260    NEXT J
270    FOR K=0 TO 5
280     MX(K)=INT((X(K)+X(K+1))/2+H)
290     MY(K)=INT((Y(K)+Y(K+1))/2+H)
300    NEXT K
310    MX(6)=MX(0) : MY(6)=MY(0)
320    FOR J=0 TO 6
330       X(J)=MX(J) : Y(J)=MY(J)
340    NEXT J
350 NEXT N
360 A$=INPUT$(1)
370 COLOR 15,4,4 : END
```



Zie de appendix voor een speciaal effect met dit programma. Het gebruik van de FNX-functie (regel 130) is hier essentieel, anders wordt het een in horizontale richting uitgerekte figuur.

**Programma 5** tekent alle diagonalen in een n-hoek. De n hoekpunten liggen op de omtrek van een ellips of van een cirkel afhankelijk van de waarden die we in het begin van het programma voor de halve lange as a en halve kleine as b kiezen. Voor de berekening van de coördinaten van de punten op de omtrek van de ellips gebruiken we niet de (cartesische) vergelijking $(x^2/a^2) + (y^2/b^2) = 1$, maar de parametervorm $x = a.\cos\varphi$ en $y = b.\sin\varphi$.

Als de parameter $\varphi$ loopt van 0° tot 360°, dan beschrijft het daaraan toegevoegde punt $P(x,y)$, met $x = a\cos\varphi$ en $y = b\sin\varphi$, precies de omtrek van een ellips. Hieronder zien we een bepaald punt $P(x_1,y_1)$ op de omtrek van de ellips met de daarbij horende waarde van de parameter $\varphi_1$. Voor dat punt geldt: $x_1 = a\cos\varphi_1$ en $y_1 = b\sin\varphi_1$.




Als $x_1 = a\cos\varphi_1$ en $y_1 = b\sin\varphi_1$ dan geldt ook

$x_1^2 = a^2\cos^2\varphi_1$ en $y_1^2 = b^2\sin^2\varphi_1$ en ook

$\frac{x_1^2}{a^2} = \cos^2\varphi_1$ en $\frac{y_1^2}{b^2} = \sin^2\varphi_1$, dus dan is

$$\frac{x_1^2}{a^2} + \frac{y_1^2}{b^2} = \cos^2\varphi_1 + \sin^2\varphi_1$$

en omdat $\cos^2\varphi_1 + \sin^2\varphi_1$ gelijk is aan 1 geldt $\frac{x_1^2}{a^2} + \frac{y_1^2}{b^2} = 1$

en hebben we de cartesische relatie tussen $x_1$ en $y_1$ afgeleid uit de parametervorm.

Voor het berekenen van de coördinaten van het j-de hoekpunt (X(J),Y(J)) verdelen we de maximale middelpuntshoek van 360° in n gelijke hoeken van elk 360/n graden.

Als A de lengte van de halve lange as is èn
B de lengte van de halve korte as èn
N het aantal hoekpunten van de n-hoek èn
W gelijk is aan 360/N èn
W1 de hoek die hoort bij het j-de hoekpunt èn
X(J) de x-coördinaat van het j-de hoekpunt t.o.v. het middelpunt van de ellips èn
Y(J) de y-coördinaat van het j-de hoekpunt t.o.v. het middelpunt van de ellips

dan zou je in een BASIC-programma de coördinaten X(J) en Y(J) als volgt kunnen berekenen:

```
W    = 360/N : W1 = J*W
X(J) = INT(A*COS(W1))
Y(J) = INT(B*SIN(W1))
```

Als we dit doen maken we twee fouten. De eerste fout is dat de hoek W1, als argument van de COSinus- en de SINusfunctie, in graden is uitgedrukt, terwijl BASIC vereist dat het argument van deze functies in radialen wordt uitgedrukt. De tweede fout is dat het HRG-systeem*) van de MSX-computer de oorsprong van het coördinatenstelsel voor het scherm in de linkerbovenhoek van het scherm verwacht en niet in het middelpunt van de ellips. Wat we dus nog moeten doen is:

a. bereken W1 in radialen èn
b. transformeer X(J) en Y(J) naar beeldschermcoördinaten.

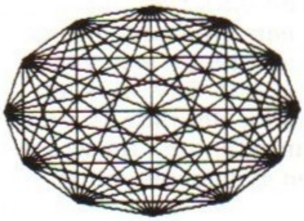
We weten dat een hoek van 360° overeenkomt met $2\pi$ radialen, waarin $\pi = 3,14159.....$ . Dit betekent dat een hoek van 1 graad overeenkomt met een hoek van $2\pi/360$ of $\pi/180$ radialen. De transformatieformules voor de transformatie van 'onze' coördinaten naar beeldschermcoördinaten hebben we op p.5 gegeven. De drie BASIC-opdrachten voor het berekenen van de beeldschermcoördinaten van het j-de hoekpunt van de n-hoek worden nu:

```
W = (360/N)*π/180 : W1 = J*W
X(J) = INT(U + A*COS(W1) + H)
Y(J) = INT(V - B*SIN(W1) + H)
```

Voor $\pi$ gebruiken we in het programma de variabele PI die als waarde krijgt 4.ATN(1). ATN is de inverse tangensfunctie. Er geldt dat tan(45°)=1, dus $\tan(\pi/4)=1$. Dit betekent dat de inverse tangens van 1 gelijk is aan $\pi/4$, dus $\text{atan}(1) = \pi/4$, hetgeen tot gevolg heeft dat $\pi = 4.\text{atan}(1)$; in BASIC: PI=4*ATN(1). We hadden ook PI=3.14159 kunnen nemen!

*) HRG = Hoge Resolutie Graphics

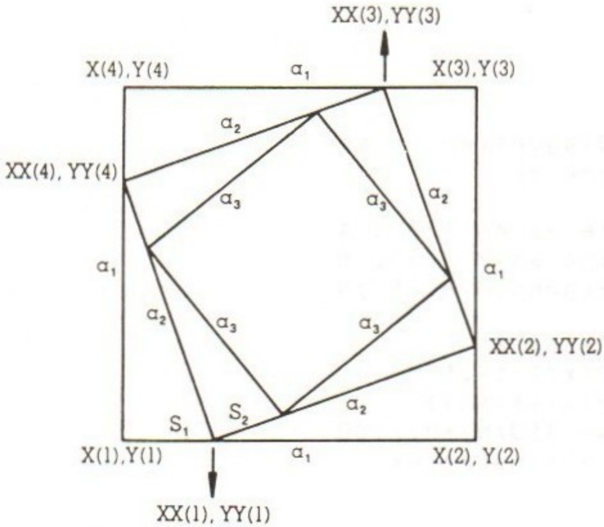
In het navolgende programma komen we bovenstaande opdrachten tegen. Kiezen we voor A en B dezelfde waarden, dan krijgen we een regelmatige n-hoek met al zijn diagonalen. Bovendien is de ellips dan een cirkel geworden.



```
100 REM programma 5 diagonalen in een
                      n-hoek
110 CLS
120 INPUT "halve grote as A<=128"; A
130 INPUT "halve kleine as B<=96"; B
140 INPUT "aantal hoekpunten N<25";N
150 CLS
160 COLOR 1,15,4 : SCREEN 2
170 DEF FNX(X)=INT(37+X/1.4+.5)
180 DIM X(N),Y(N) : PI=4*ATN(1)
190 U=128:V=96:H=.5:W=(360/N)*PI/180
200 REM punten op de ellipsomtrek
        berekenen
210 FOR J=0 TO N-1
220    W1=J*W
230    X(J)=INT(U+A*COS(W1)+H)
240    Y(J)=INT(V-B*SIN(W1)+H)
250 NEXT J
260 REM punten op de ellipsomtrek
        met elkaar verbinden
270 FOR I=0 TO N-2
280    FOR J=I+1 TO N-1
290     LINE (FNX(X(I)),Y(I))-
             (FNX(X(J)),Y(J))
300    NEXT J
310 NEXT I
320 A$=INPUT$(1)
330 COLOR 15,4,4 : END
```

Programma 6 is wat veeleisender in die zin dat het wat voorbereiding nodig heeft.

We willen een reeks ingeschreven vierkanten zo tekenen dat de hoekpunten van een ingeschreven vierkant uit de reeks op de zijden liggen van zijn voorganger. We zien deze situatie voor het eerste, tweede en derde vierkant van de reeks in de onderstaande tekening. De afstand tussen een hoekpunt van een vierkant en een hoekpunt van het volgende vierkant ($S_1$ en $S_2$) is steeds een vast deel van de zijde waarop de hoekpunten liggen. Zo is $S_1 = a_1/k$ en $S_2 = a_2/k$. In het algemeen is $S_n = a_n/k$, waarbij $a_n$ de zijde is van het n-de vierkant in de reeks en k de verkleiningsfactor. De waarde voor k kunnen we zelf kiezen. k=16 geeft een fraaie tekening.




(X(1),Y(1)); (X(2),Y(2)); (X(3),Y(3)) en (X(4),Y(4)) zijn de coördinaten van de hoekpunten van een bepaald vierkant uit de reeks. Hoe berekenen we nu de coördinaten van de hoekpunten van het volgende ingeschreven vierkant ((XX(1),YY(1)); (XX(2),...)?

Voor het eerste hoekpunt van het eerste ingeschreven vierkant geldt:

$$XX(1) = X(1) + \frac{X(2)-X(1)}{K} \quad \text{èn}$$

$$YY(1) = Y(1) + \frac{Y(2)-Y(1)}{K}$$

Ga na dat dit klopt in bovenstaande tekening.

Voor het tweede hoekpunt van het ingeschreven vierkant geldt:

$$XX(2) = X(2) + \frac{X(3)-X(2)}{K} \quad \text{èn}$$

$$YY(2) = Y(2) + \frac{Y(3)-Y(2)}{K}$$

In het algemeen geldt:

$$XX(J) = X(J) + \frac{X(J+1)-X(J)}{K} \quad \text{èn}$$

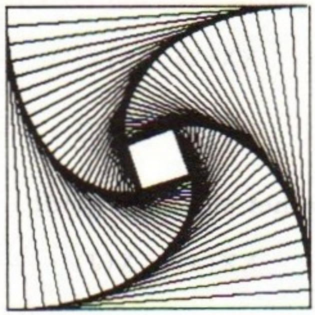
$$YY(J) = Y(J) + \frac{Y(J+1)-Y(J)}{K}$$

voor J = 1,2,3,4, waarbij X(5) = X(1) en
Y(5) = Y(1).

Als we het tweede vierkant getekend hebben, veranderen we XX(1) in X(1), YY(1) in Y(1), XX(2) in X(2), YY(2) in Y(2), enzovoorts, en tenslotte X(5) in X(1) en Y(5) in Y(1) en we gaan met dezelfde formules opnieuw XX(1),YY(1),... enz. berekenen, maar dan voor de hoekpunten van het volgende vierkant. We zien dit gebeuren in de regels 270 t/m 300 van het volgende programma. De regels 200 t/m 220 tekenen het vierkant, terwijl de regels 230 t/m 260 de hoekpunten van het volgende vierkant berekenen. Een structuurdiagram voor het programma ziet er zo uit:

<table>
<tr><td colspan="2">begin programma ingeschreven vierkanten</td></tr>
<tr><td colspan="2">HRG-voorbereidingen</td></tr>
<tr><td colspan="2">lees waarde voor K in</td></tr>
<tr><td colspan="2">bepaal coördinaten voor het eerste vierkant</td></tr>
<tr><td colspan="2">voor het eerste t/m 40-ste vierkant doe</td></tr>
<tr><td rowspan="2"></td><td>teken dit vierkant</td></tr>
<tr><td>bepaal coördinaten voor het volgende vierkant</td></tr>
<tr><td colspan="2">einde programma</td></tr>
</table>

Hier komt het programma:



```
100 REM programma 6 ingeschreven
                     vierkanten
110 CLS
120 INPUT "toets K in 1<K<20"; K
130 COLOR 1,15,4 : SCREEN 2
140 DIM X(5),Y(5),XX(5),YY(5)
150 FOR J=1 TO 5
160    READ X(J),Y(J)
170 NEXT J
180 CLS : H=.5
190 FOR N=1 TO 40
200    FOR J=1 TO 4
210       LINE (X(J),Y(J))-
               (X(J+1),Y(J+1))
220    NEXT J
230    FOR J=1 TO 4
240       XX(J)=X(J)+
               INT((X(J+1)-X(J))/K+H)
250       YY(J)=Y(J)+
               INT((Y(J+1)-Y(J))/K+H)
260    NEXT J
270    FOR J=1 TO 4
280       X(J)=XX(J) : Y(J)=YY(J)
290    NEXT J
300    X(5)=X(1) : Y(5)=Y(1)
310 NEXT N
320 A$=INPUT$(1)
330 COLOR 15,4,4 : END
340    DATA 60,192,196,192,196,0,60,0
          ,60,192
```

U kunt dit programma als uitgangspunt voor een uitgebreider programma gebruiken. Verdeel het beeldscherm in, bijvoorbeeld, twee rijen van vijf gelijke vierkanten (elk vierkant 50x50 puntjes). In elk van deze tien vierkanten tekent u, met bovenstaand programma, een reeks van ingeschreven vierkanten. Teken tien identieke reeksen, of kies steeds een andere waarde voor K of probeer ze ten opzichte van elkaar te laten draaien.

# 2 Grafieken van functies in cartesische vorm

In het eerste hoofdstuk hebben we alleen rechtlijnige patronen getekend, bijvoorbeeld het programma voor het tekenen van alle diagonalen in een regelmatige n-hoek.

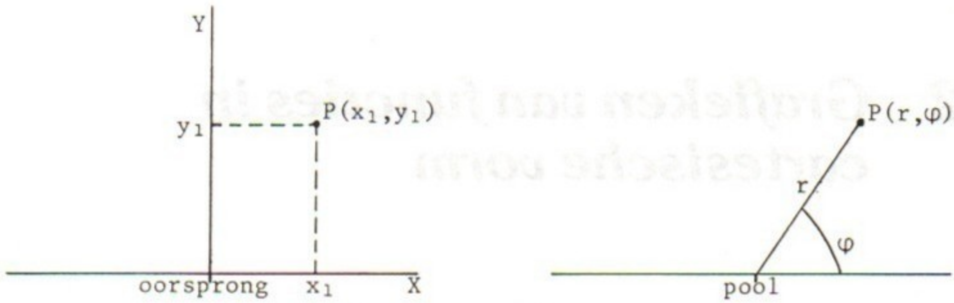
In dit en het volgende hoofdstuk zullen we ons uitvoerig bezig houden met het tekenen van de grafiek van een aantal continue en niet-continue functies. In de wiskunde noemen we de grafiek van een niet-lineaire functie een kromme. De grafiek van een lineaire functie (bijvoorbeeld de functie $y = 4x + 3$) noemen we een rechte. We kunnen de vergelijking van zo'n kromme op drie manieren formuleren:

| | | |
|---|---|---|
| A ; | met cartesische coördinaten | : $y = f(x)$ |
| B ; | met poolcoördinaten | : $r = f(\varphi)$ |
| C ; | of in parametervorm | : $x = f(t)$, $y = g(t)$ |

Niet-wiskundigen kennen vaak alleen de gewone (cartesische) vorm $y = f(x)$. Een voorbeeld van een niet-lineaire functie in cartesische vorm is de functie $y = 2x^2 - 3x + 4$. De grafiek van deze functie is een parabool.

De functie $r = 110.\cos(4\varphi)$ stelt een kromme in poolcoördinaten voor. Als we de hoek $\varphi$ laten lopen van 1 tot 360 graden en we tekenen bij elke hoek $\varphi$ onder die hoek een punt op afstand r van de oorsprong, dan krijgen we een soort bloem als op pagina 33. Deze vorm $r = 110.\cos(4\varphi)$ heet de poolcoördinatenvorm. Een punt in het platte vlak wordt nu niet gekenmerkt door de afstand van dat punt tot de x- en y-as (cartesisch), maar door de afstand tot de oorsprong (r) en de hoek $\varphi$ die de x-as maakt met de lijn die dat punt met de oorsprong verbindt. Poolcoördinaten komen in hoofdstuk 3 aan de orde (zie tekening op p.16).

Er is nog een manier om de vergelijking van een kromme weer te geven en dat is de parametervorm. Hierbij worden de x- en y-coördinaat van een punt op de kromme afhankelijk gemaakt van een




Hetzelfde punt P in een
cartesisch coördinatenstelsel en een poolcoördinatenstelsel

bepaalde parameter. Een voorbeeld zijn de vergelijkingen

$$x = a.\cos(t) \quad \text{en} \quad y = a.\sin(t).$$

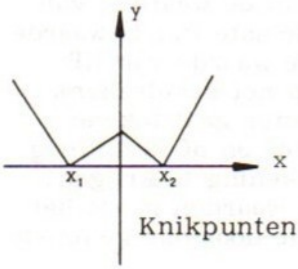
Als we t laten lopen van 0 tot 360°, beschrijven de daarbij horende punten (x,y) precies de omtrek van een cirkel met straal a. De cartesische vorm van deze cirkelvergelijking is $x^2 + y^2 = a^2$, die velen direct zullen herkennen als de 'cirkelvergelijking'.

In dit hoofdstuk houden we ons dus alleen bezig met het tekenen van grafieken van functies en relaties die door middel van cartesische coördinaten beschreven worden. De andere twee vormen komen in hoofdstuk 3 aan de orde.

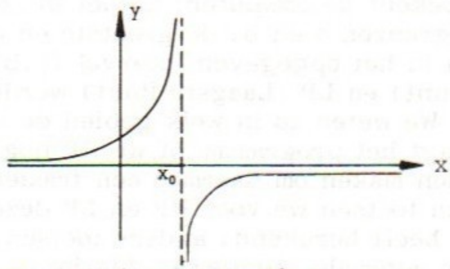
## Continue functies

Een continue functie is een functie waarvan de grafiek in één vloeiende beweging van ons 'potlood', dat wil zeggen zonder het potlood van het papier te hoeven halen, getekend kan worden. Er mogen 'knikken' in voorkomen maar geen onderbrekingen (zie tekening op p.17).

De linkergrafiek kunnen we in één beweging, zonder het potlood van het papier te halen, tekenen; bij de rechter grafiek kan dat niet. De linker grafiek is de grafiek van een continue functie; de rechter van een niet-continue functie. Als we alleen links of alleen rechts van het punt $x_0$ kijken dan is de functie op die intervallen natuurlijk wel continu! De functie is alleen niet-continu in het punt $x_0$.




Knikpunten




Functie is niet-continu
in het punt $x_0$

We willen nu een algemeen programma ontwikkelen dat, gegeven de grenzen a en b van een bepaald interval ($a \leq x \leq b$) de grafiek tekent van een willekeurige continue functie y = f(x). Mochten de x-as en de y-as in het gebied liggen waarin de grafiek van de functie wordt getekend, dan willen we dat ook beide assen door het programma getekend worden. Om het programma kort, maar toch algemeen, te houden brengen we de volgende vereenvoudigingen aan.

1. De vergelijkingen van de functie, die getekend moet worden, wordt in een subroutine, vanaf regel 1000, opgenomen. De functiewaarde y = f(x), bij een bepaalde x-waarde, wordt berekend door op dat moment met de gewenste x-waarde de subroutine aan te roepen (GOSUB 1000). Er wordt dus niet gebruik gemaakt van de methode waarin de functie als tekst (INPUT-opdracht met stringvariabele) wordt ingelezen waarna het programma deze tekst zelf omzet in een goede BASIC functiedefinitie. Een andere mogelijkheid zou zijn om de functie met een DEF FN-opdracht in het programma te definiëren.

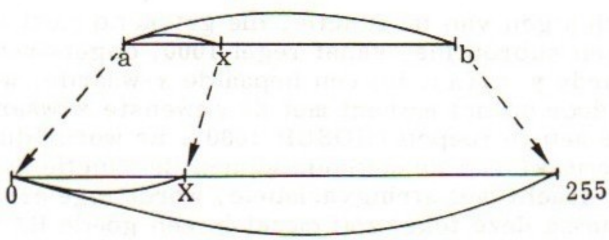
2. Op de coördinaatassen wordt niet automatisch een schaalverdeling aangebracht en ook wordt er geen tekst bij afgedrukt. Via een omweg kan uw MSX-computer wel tekst op het hoge-resolutiescherm afdrukken. We geven hiervan in de appendix twee voorbeelden. Kijk vast in de appendix als u naast tekeningen ook tekst en getalwaarden wilt afdrukken. Wilt u toch graag weten wat bij een bepaalde waarde van x de functiewaarde (y) is, dan kunt u bijvoorbeeld na het tekenen van de grafiek de computer (op de printer) een lijstje met x- en y-waarden laten afdrukken.

Terug naar het programma-ontwerp. In het eerste deel van het programma berekent de computer, na het inlezen van de waarden van de intervalgrenzen a en b, de grootste en de kleinste functiewaarde (y-waarde) in het opgegeven interval [a,b]. De waarde van HP (Hoogste Punt) en LP (Laagste Punt) worden op het beeldscherm afgedrukt. We weten zo in welk gebied de computer gaat tekenen. Hierna vraagt het programma of we HP nog groter en of we LP nog kleiner willen maken om daarmee een fraaiere tekening te krijgen. Zo niet, dan toetsen we voor HP en LP dezelfde waarden in als het programma heeft berekend; anders toetsen we de door ons gewenste maximale en minimale functiewaarden in.

Het volgende programmadeel (regels 290-340) bevat de lus (FOR X=A TO B STEP DX) voor het tekenen van de grafiek. De coördinaten x,y van een punt op de grafiek moeten met de juiste transformatieformules omgezet worden in beeldschermcoördinaten X,Y.

Om het interval [a,b] voor te kiezen x-waarden ($a \leq x \leq b$) om te zetten in het HRG-interval [0,255] ($0 \leq X \leq 255$) gebruiken we de volgende evenredigheid:

$$(x-a) : (b-a) = X : 255 \quad \text{ofwel} \quad \frac{x-a}{b-a} = \frac{X}{255}$$




in BASIC:

```
XX = INT(KX*(X-A)+0.5),    met KX = 255/(B-A).
```

Omdat BASIC in hoofdletters 'werkt' hebben we in deze formule voor x de variabele X genomen, voor X de variabele XX, voor a de variabele A en voor b de variabele B.

We doen dit ook voor het afbeelden van het functiebereik $LP \leq y \leq HP$ op het HRG-beeldbereik $0 \leq Y \leq 191$:

$$(HP-y) : (HP-LP) = Y : 191 \quad \text{ofwel} \quad \frac{HP-y}{HP-LP} = \frac{Y}{191}$$

Dit geeft in BASIC:

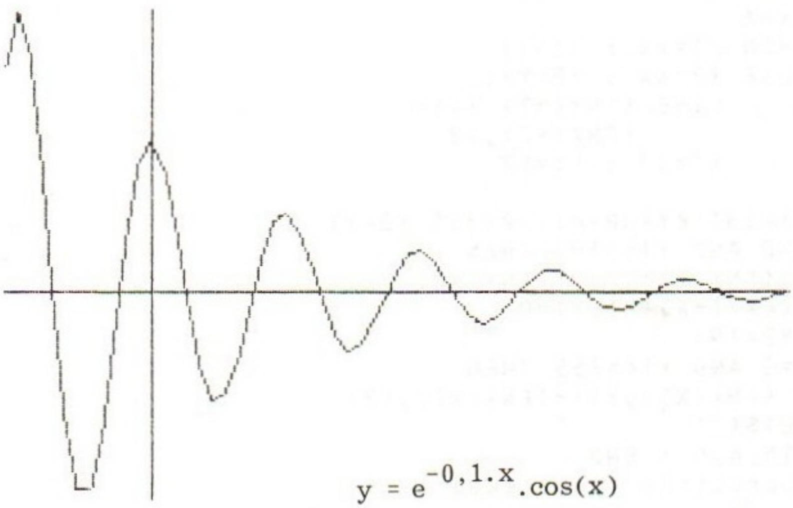
```
YY = INT(KY*(HP-Y)+0.5)   met  KY = 191/(HP-LP)
```

We zien deze berekeningen van XX en YY in de regels 310 en 320. In de volgende regels worden twee, op deze wijze berekende, punten (X1,Y1) en (X2,Y2) door een recht lijntje met elkaar verbonden. Vervolgens (regels 350-380) wordt de ligging van de x- en y-as bepaald. Kiezen we in bovenstaande transformatievergelijkingen voor X en Y de waarde nul, dan vinden we respectievelijk de ligging van de y-as en van de x-as (regels 350-370).

Met deze uitleg is de werking van het programma hopelijk te volgen.

Test u het programma 7 met willekeurige, maar continue, functies. In dit voorbeeldprogramma hebben we de functie $y = e^{-0.1x}.\cos(x)$ gekozen, hetgeen de grafiek van een gedempte trilling oplevert. Hieronder zijn enkele ideeën voor minder moeilijke functies.

| functie | regel 1000 | waarde voor a | waarde voor b |
|---|---|---|---|
| $y = \sin(x)$ | Y = SIN(X) | 0 | 2π (= 6,2832) |
| $y = x^2$ | Y = X∧2 | -4 | +4 |
| $y = e^x$ | Y = EXP(X) | -3 | +3 |
| $y = x^3-2x^2-x$ | Y = X∧3-2*X∧2-X | -1 | +3 |



$y = e^{-0,1.x}.\cos(x)$

(zie regel 1010 programma 7 op p.20)

```
100 REM programma 7 grafiek van een
                      continue functie
110 CLS
120 INPUT "linker intervalgrens "; A
130 INPUT "rechter intervalgrens"; B
140 IF A>B THEN SWAP A,B
150 HP=-100000!:LP=100000!:
    DX=(B-A)/64
160 FOR X=A TO B STEP DX
170    GOSUB 1000 'functiewaarde
                      berekenen
180    IF Y>HP THEN HP=Y
190    IF Y<LP THEN LP=Y
200 NEXT X
210 PRINT "grootste Y-waarde:"; HP
220 PRINT "kleinste Y-waarde:"; LP
230 INPUT "bovengrens voor Y"; HP
240 INPUT "ondergrens voor Y"; LP
250 CLS
260 COLOR 1,15,4 : SCREEN 2
270 DEF FNX(X)=INT(37+X/1.4+.5)
280 KX=255/(B-A):KY=191/(HP-LP):H=.5
290 FOR X=A TO B STEP DX
300    GOSUB 1000 'functiewaarde
                     berekenen
310    XX=INT(KX*(X-A)+H)
320    YY=INT(KY*(HP-Y)+H)
330    IF X=A
         THEN X1=XX : Y1=YY
         ELSE X2=XX : Y2=YY:
              LINE (FNX(X1),Y1)-
                   (FNX(X2),Y2):
              X1=X2 : Y1=Y2
340 NEXT X
350 X1=0:Y1=INT(KY*HP+H):X2=255:Y2=Y1
360 IF Y1>=0 AND Y1<=191 THEN
      LINE (FNX(X1),Y1)-(FNX(X2),Y2)
370 X1=INT(KX*(-A)+H):Y1=0:
    X2=X1:Y2=191
380 IF X1>=0 AND X1<=255 THEN
      LINE (FNX(X1),Y1)-(FNX(X2),Y2)
390 A$=INPUT$(1)
400 COLOR 15,4,4 : END
1000 REM subroutine functiewaarde
                      berekenen
1010  Y=EXP(-.1*X)*COS(X)
1020 RETURN
```

Nu volgen drie korte programma's.

Programma 8 is de eerste van deze drie. Dit programma tekent tien in fase verschoven sinuskrommen, allemaal in hetzelfde coördinatenstelsel. De vergelijking van het stelsel is

$$y = \sin(x+np), \quad \text{met als fase } p = \frac{\pi}{9} \ (20°).$$

We vinden de tien vergelijkingen door voor n de waarden 0,1,2 t/m 9 te kiezen. De lusvariabele J (regel 160) komt overeen met de beeldschermcoördinaat X2. Omdat we in dit programma de beeldschermcoördinaat X2, dat wil zeggen J, in stappen van 5 naar 255 laten lopen (regel 160), moeten we eerst de bij X2 behorende waarde voor x berekenen. Dan pas kunnen we de functiewaarde y = f(x) berekenen. We kunnen de evenredigheid

$$(x-a) : (b-a) = X2 : 255$$

natuurlijk ook gebruiken om x uit X2 te berekenen; hieruit volgt:

$$x = \frac{(b-a)}{255}.X2 + a.$$

In het onderstaande programma zijn de gekozen intervalgrenzen a = 0 en b = 2π, zodat bovenstaande vergelijking wordt:

$$x = \frac{2\pi}{255}.X2.$$

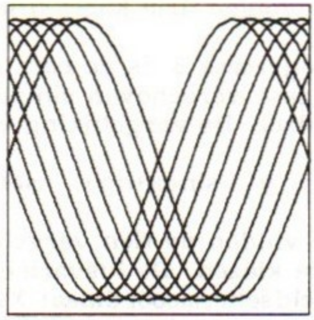
Als we $C = \frac{2\pi}{255}$ nemen, wordt in het programma het argument van de sinusfunctie dus C*X2+N*P, ofwel C*J+N*P.

In het programma wordt de waarde C*J als X berekend, dus zien we in regel 170 X+N*P als argument van de SINusfunctie. Natuurlijk moeten we de functiewaarde y = SIN(X+N*P) nog omzetten naar de beeldschermcoördinaat Y zodat we krijgen

Y2 = INT(V-K*SIN(X+N*P)+H).

Door voor de schaalconstante K de waarde 90 te kiezen (regel 140) krijgen we een mooie 'grote' tekening.

Wie met kleuren wil werken kan dit programma uitbreiden door de diverse krommen andere kleuren te geven. Iets moeilijker zal het zijn de 'banen' tussen de sinuskrommen in te kleuren. Probeer het eens.



```
100 REM programma 8 sinuskrommen
110 CLS : PI= 4*ATN(1)
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 V=96:K=90:H=.5:P=PI/9:C=2*PI/255
150 FOR N=0 TO 9
160  FOR J=0 TO 255 STEP 5
170   X=J*C :
      Y=INT(V-K*SIN(X+N*P)+H)
180   IF J=0
       THEN X1=J:Y1=Y
       ELSE
        X2=J:Y2=Y:
        LINE(FNX(X1),Y1)-(FNX(X2),Y2)
       :X1=X2:Y1=Y2
190  NEXT J
200 NEXT N
210 A$=INPUT$(1)
220 COLOR 15,4,4 : END
```

MSX kleurcodes

| Code | kleur | code | kleur |
|---|---|---|---|
| 0 | transparant | 8 | middelrood |
| 1 | zwart | 9 | lichtrood |
| 2 | middelgroen | 10 | donkergeel |
| 3 | lichtgroen | 11 | lichtgeel |
| 4 | donkerblauw | 12 | donkergroen |
| 5 | lichtblauw | 13 | magenta |
| 6 | donkerrood | 14 | grijs |
| 7 | hemelsblauw | 15 | wit |

**Programma 9** tekent een stelsel parabolen met de vergelijking

$y = -tx^2 + t$

Alle parabolen (bepaalde t-waarden) hebben dezelfde snijpunten met de x-as (x = 1 en x = -1).



```
100 REM programma 9 paraboolstelsel
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 U=128:V=96:H=.5
150 FOR K=- 90 TO 90 STEP 10
160     FOR X=-110 TO 110 STEP 5
170       XX=INT(U+X+H)
180       Y=-K*X*X/6400+K:Y=INT(V-Y+H)
190        IF X= -110
             THEN X1=XX:Y1=Y
             ELSE X2=XX:Y2=Y:
                  LINE (FNX(X1),Y1)-
                       (FNX(X2),Y2):
                  X1=X2:Y1=Y2
200     NEXT X
210 NEXT K
220 A$=INPUT$(1)
230 COLOR 15,4,4 : END
```

Voeg toe:
```
165 IF X<=0 THEN KLEUR=10
            ELSE KLEUR=4
```

en verander de LINE-opdracht in regel 190 in:

```
LINE(FNX(X1),Y1)-
(FNX(X2),Y2),KLEUR:
```

Ook heel mooi wordt het met:

```
165 IF INT((K/10)/2)*2=K/10
              THEN KLEUR=10
              ELSE KLEUR=5
```

Soms willen we de oppervlakte tussen de grafiek van een functie en de x-as berekenen (de integraal bepalen) of laten tekenen.

Programma 10 kleurt zo'n oppervlak tussen de x-as en de grafiek van de functie

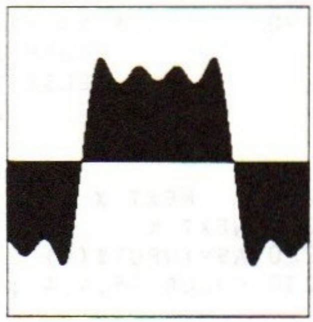
$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7}.$$

Ook nu is de lusvariabele J de HRG-coördinaat X2. De waarde voor a is $-\pi$ en die voor b is $+\pi$, waaruit volgt dat

$$x = \frac{J.2\pi}{255} - \pi.$$

Ook nu nemen we in het programma $C = \frac{2\pi}{255}$

```
100 REM programma 10  oppervlak onder
                      een kromme
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 V=96:H=.5:PI=4*ATN(1):K=80:
    C=2*PI/255
150 FOR J=0 TO 255
160    X=J*C-PI
170    GOSUB 1000 'functiewaarde
                   berekenen
180    Y=INT(V-K*Y+H)
190    LINE (FNX(J),V)-(FNX(J),Y)
200 NEXT J
210 A$=INPUT$(1)
220 COLOR 15,4,4 : END
1000 REM subroutine functewaarde
                    berekenen
1010 Y=COS(X)-COS(3*X)/3+COS(5*X)/5-
     COS(7*X)/7
1020 RETURN
```



Probeer dit programma zo te maken dat de drie oppervlakken in bijvoorbeeld rood, wit en blauw gekleurd worden. Gebruik hiertoe de mogelijkheid om in de LINE-opdracht de afdrukkleur van de lijn op te geven (zie p.23).

## Niet-overal-continue functies

De functie $y = \frac{x^2+3}{x^2-x-6}$ kan niet met behulp van programma 7 getekend worden. Zouden we bijvoorbeeld voor x het interval $-5 \leq x \leq 5$ kiezen, dan liggen hierin de punten x = 3 en x = -2. Dit zijn de twee punten waarvoor de noemer van bovenstaande functie nul is. Als de computer bij het tekenen bij één van deze twee punten belandt, zal op het scherm een foutmelding (DIVISION BY ZERO) 'delen door nul' verschijnen en zal het programma afgebroken worden.

Dit zal ook gebeuren bij logaritmische functies met een niet-positief argument of bij wortelfuncties met een negatief argument. Zelfs bij het tekenen van continue (nette) functies kunnen problemen optreden. We komen hierbij in de problemen als de functiewaarden heel groot of heel klein worden. Transformatie van dergelijke waarden naar HRG-beeldschermcoördinaten (0-255 en 0-191) heeft dan geen enkele zin meer, omdat de aard van het verloop van de functie in het geheel niet meer tot uitdrukking komt. Het is daarom beter om de functiewaarden van een continue functie naar boven en beneden te begrenzen. Dit heeft tot gevolg dat de grafiek van een continue functie er uit zou kunnen zien als de grafiek van een niet-continue functie.

Het zal duidelijk zijn dat je een algemeen programma voor het tekenen van een willekeurige continue of niet-continue functie niet zomaar even opschrijft. Dergelijke programma's kom je in de vakliteratuur dan ook niet of nauwelijks tegen, en mocht je er wel een tegenkomen, dan betreft het òf een heel groot programma òf een programma dat voor een bepaalde functie, waarvan men van te voren weet waar de discontinuïteiten zitten, geschreven is.

We geven nu een verbazend kort programma voor het tekenen van de grafiek van een willekeurige functie y = f(x). Veel wiskundeleraren, scholieren en studenten zullen nu hun 'oren' spitsen. U begrijpt dat, gezien het bovenstaande, hierbij wel enkele beperkingen gelden:

1. Degene die het programma gaat gebruiken moet een beetje kunnen programmeren. De functie moet namelijk in gedeelten in een subroutine (vanaf regel 1000) beschreven worden.
2. De programmagebruiker moet voldoende wiskundige kennis bezitten om te kunnen bepalen voor welke x-waarden functies moeilijkheden kunnen opleveren.

We zullen zien dat, normaal gesproken, slechts een paar BASIC-regels nodig zijn om de functiebeschrijving te programmeren. Het berekenen van nulpunten met behulp van een of ander numeriek-wiskundig algoritme is in het geheel niet nodig.

In het onderstaande programma 11 gebruiken we twee variabelen FZ en FA als zogeheten vlaggen (Flags). Een vlag is een variabele die slechts twee waarden (vaak 0 en 1) kan aannemen.

Om de werking van de vlag FZ duidelijk te maken geven we hieronder de subroutine 1000 met de functiebeschrijving van de functie

$$y = \frac{x^2+3}{x^2-x-6}$$

```
REM subroutine functiewaarde
                berekenen
 N=X*X-X-6
 IF N=0 THEN FZ=1
        ELSE Y=(X*X+3)/N:
             IF Y<LP OR Y> HP
               THEN FZ=1
               ELSE FZ=0
RETURN
```

De vlag FZ wordt alleen op 1 gezet als het punt (x,y) niet op het scherm getekend kan worden. Dit is het geval als de functie niet-continu is in het punt (x,y) (N = 0; x = 3 en x = -2) of als de functiewaarde buiten het opgegeven functiebereik ($LP \leq y \leq HP$) valt.

Waarvoor dient nu de tweede vlag FA? Bekijk het volgende stukje programma eens:

```
FA=1
FOR X=A TO B STEP DX
 X2=INT(KX*(X-A)+H)
 GOSUB 1000 'functiewaarde
              berekenen
 IF FZ=1 THEN FA=1
         ELSE IF FA=1
  THEN X1=X2:FA=0:
   Y1=INT(KY*(HP-Y)+H)
  ELSE Y2=INT(KY*(HP-Y)+H):
   LINE(FNX(X1),Y1)-
   (FNX(X2),Y2):X1=X2:Y1=Y2
NEXT X
```

Als FZ = 1 dan wordt FA ook gelijk aan 1 gemaakt en wordt de volgende X-waarde bekeken (FOR-lus) zonder dat het nieuw berekende punt met het laatstgetekende punt verbonden wordt. Is FZ echter 0 (berekende punt ligt in het beeldvlak) dan wordt onderzocht of FA daarvoor soms op 1 gezet is. Is dit zo (tweede THEN) dan moet het nieuw berekende punt als nieuw beginpunt (X1,Y1) gekozen worden en dit mag dan ook niet met het vorige getekende punt verbonden worden. FA wordt in dit geval nul gemaakt.

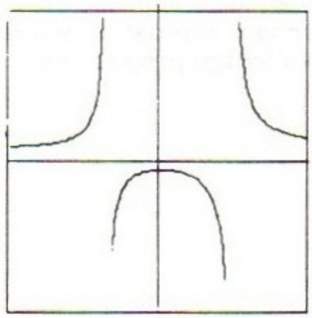
Dus als zowel FZ als FA nul zijn, wordt het nieuw berekende punt verbonden met het laatstgetekende punt.

Om te zorgen dat het programma probleemloos werkt moet voor het berekenen van het eerste punt van de grafiek, dus voor de FOR-lus, de vlag FA op 1 gezet worden, waardoor we er zeker van zijn dat de waarden X1 en Y1 berekend worden. Hier volgt het volledige programma:

```
100 REM programma 11 grafiek van een
         willekeurige functie
110 CLS
120 INPUT "linkergrens voor X  "; A
130 INPUT "rechtergrens voor X "; B
140 INPUT "ondergrens voor Y   "; LP
150 INPUT "bovengrens voor Y   "; HP
160 IF A>B THEN SWAP A,B
170 KX=255/(B-A):KY=191/(HP-LP):H=.5
180 DX=(B-A)/255
190 CLS
200 COLOR 1,15,4 : SCREEN 2
210 DEF FNX(X)=INT(37+X/1.4+.5)
220 FA=1
230 FOR X=A TO B STEP DX
240  X2=INT(KX*(X-A)+H)
250  GOSUB 1000 'functiewaarde
                 berekenen
260  IF FZ=1 THEN FA=1
              ELSE IF FA=1
       THEN X1=X2:FA=0:
        Y1=INT(KY*(HP-Y)+H)
       ELSE Y2=INT(KY*(HP-Y)+H):
        LINE(FNX(X1),Y1)-
        (FNX(X2),Y2):X1=X2:Y1=Y2
270 NEXT X
280 X1=0:Y1=INT(KY*HP+H):X2=255:Y2=Y1
290 IF Y1>0 AND Y1<=191  THEN
      LINE (FNX(X1),Y1)-(FNX(X2),Y2)
300 X1=INT(KX*(-A)+H):Y1=0:X2=X1:
    Y2=191
310 IF X1>0 AND X1<=255  THEN
      LINE (FNX(X1),Y1)-(FNX(X2),Y2)
320 A$=INPUT$(1)
330 COLOR 15,4,4 : END
1000 REM subroutine functiewaarde
                    berekenen
1010  N=X*X-X-6
1020  IF N=0 THEN FZ=1
              ELSE Y=(X*X+3)/N:
                   IF Y<LP OR Y> HP
                     THEN FZ=1
                     ELSE FZ=0
1030 RETURN
```

Dit programma tekent de grafiek van de functie

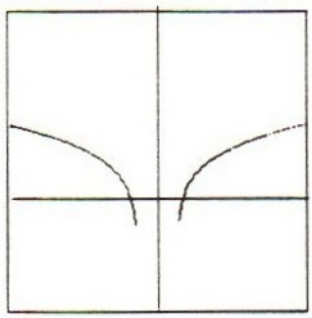
$y = \frac{x^2+3}{x^2-x-6}$ ; kies voor A, B, LP en HP respectievelijk -5, 5, -10 en 10



Wilt u een andere functie tekenen, bijvoorbeeld $y = \ln(x^2-2)$, herschrijf dan subroutine 1000 als volgt:

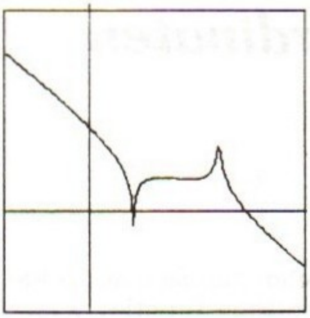
```
1000 REM subroutine functiewaarde
                    berekenen
1010  N=X*X-2
1020  IF N<=0 THEN FZ=1
              ELSE Y=LOG(N):
                   IF Y<LP OR Y> HP
                   THEN FZ=1
                   ELSE FZ=0
1030 RETURN
```

U krijgt dan deze grafiek:



Kies voor A, B, LP en HP de waarden -25, 25, -5 en 8.

Hier volgt een hele fraaie:



$y=3-x+\ln\left|\frac{x-1}{x-3}\right|$

Subroutine 1000 wordt nu:

```
1000 REM subroutine functiewaarde
                    berekenen
1010  N=X-3
1020  IF N=0 THEN FZ=1
          ELSE Y=3-X+LOG(ABS((X-1)
                              /N)):
                IF Y<LP OR Y> HP
                    THEN FZ=1
                    ELSE FZ=0
1030 RETURN
```

Kies voor A, B, LP en HP de waarden -5, 10, -4 en 8.

# 3 Krommen in poolcoördinaten en in parametervorm

In het vorige hoofdstuk hebben we ons beziggehouden met het tekenen van de grafiek van een continue of niet-continue functie, waarvan de vergelijking als y = f(x) geschreven kon worden; dus in cartesische vorm. Nu gaan we grafieken tekenen van functies waarvan de vergelijking in poolcoördinaten geschreven wordt of in de zogeheten parametervorm. Dit levert zeer fraaie 'beelden' op.

## Krommen met poolcoördinaten

Functies, waarvan de grafiek een gesloten kromme laat zien, zijn vaak eenvoudiger met poolcoördinaten te beschrijven dan in cartesische vorm. Als voorbeeld noemen we de 'hartkromme' (kardioide), waarvan de vergelijking in poolcoördinaten eenvoudig is, namelijk:

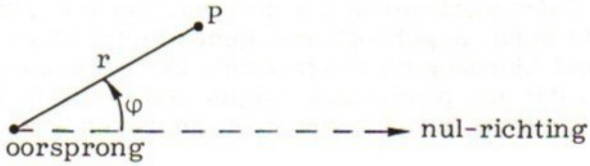
$$r = k(1 + \cos\varphi),$$

maar waarvan de cartesische vorm nogal ingewikkeld is, namelijk

$$\frac{(x^2-y^2-kx)^2}{k^2(x^2+y^2)} = 1.$$

Om de volgende programma's te doorgronden (niet om ze te draaien!) is een beetje theorie nodig. Het poolcoördinatenstelsel wordt in de wiskundelessen op school niet of nauwelijks behandeld. Eigenlijk is dit jammer, want hiermee (en met de parametervorm) kunnen juist de mooiste functies (en daarmee de fraaiste grafieken) beschreven en getekend worden.

In een poolcoördinatenstelsel wordt elk punt in het platte vlak met twee coördinaten, te weten r en $\varphi$, bepaald. r is de afstand tussen het punt en de oorsprong en $\varphi$ (phi, spreek uit: fie) is de hoek tussen de horizontale lijn door de oorsprong en de lijn door de oorsprong en het punt (zie tekening op p.31).

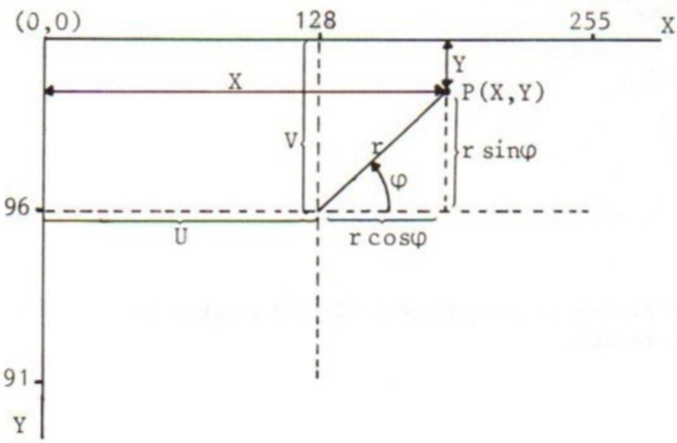



De oorsprong van het poolcoördinatenstelsel leggen we in het middelpunt van het beeldscherm (U = 128, V = 96). Zoals we weten ligt de oorsprong van het HRG-beeldschermcoördinatenstelsel in de linkerbovenhoek van het beeldscherm (HOME-positie). De nul-richting in ons poolcoördinatenstelsel is horizontaal en wijst naar rechts. $\varphi=90°$ ($\pi/2$ radialen) is verticaal naar boven; $\varphi=180°$ ($\pi$ radialen) is horizontaal naar links en $\varphi=270°$ ($3\pi$ radialen) verticaal naar beneden.

We gebruiken de volgende twee transformatieformules om het, uit de functievergelijking berekende, punt P(r,φ) om te zetten in het beeldschermpunt P(X,Y):

```
X = INT(U + R*COS(P) + 0.5)      èn
Y = INT(V - R*SIN(P) + 0.5)
```

In onderstaande figuur zien we hiervoor de verklaring.

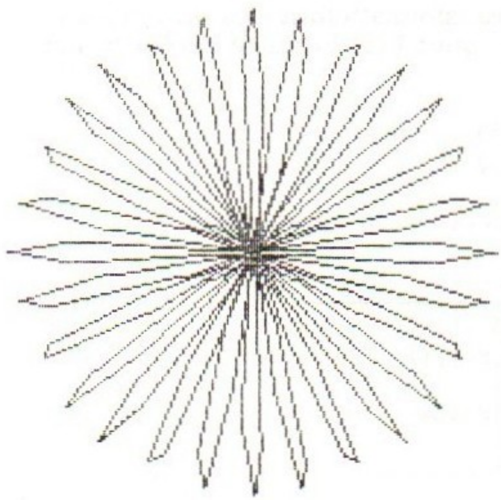



De hoek φ, in de programma's voorgesteld door de variabele P, doorloopt steeds in positieve richting (tegen de klok in) het interval van 0 tot $2\pi$. Dit is een interval in radialen (0-360 in graden).

In alle programma's is als lusvariabele de hoek W in graden gekozen. Met de formule P = W.($\pi$/180) (in de programma's P = W*RD) wordt de hoek in radialen berekend. Het voordeel van een lusvariabele

die het interval 0-360° doorloopt is dat de stapgrootte waarmee de lusvariabele steeds wordt opgehoogd een geheel getal kan zijn (1 bijvoorbeeld) en dat hierdoor de programma's beter leesbaar zijn. Een neveneffect is dat alle programma's bijna woordelijk in Pascal zijn over te zetten; Pascal kent immers geen gebroken stapgrootte in een lus.

Meer theorie hebben we niet nodig. De volgende programma's spreken grotendeels voor zichzelf.

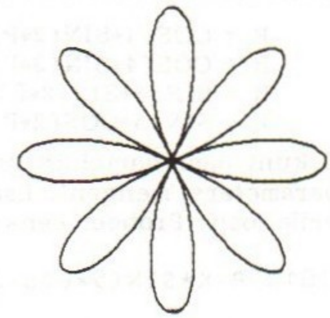


Deze figuur ontstaat als we in programma 12 (zie pagina hiernaast) voor N de waarde 14 nemen.

Programma 12 tekent de grafiek van de functie

r = k cos(nφ).

In het programma kiezen we k = 95 (K) en n = 4 (N).



```
100 REM programma 12 grafiek van de
        functie R=K*cos(4*Phi)
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 PI=4*ATN(1):N=4
150 U=128:V=96:H=.5:RD=PI/180:K=95
160 FOR W=0 TO 360 STEP 3
170     P=W*RD
180     GOSUB 1000 'functiewaarde
                   berekenen
190     X=INT(U+R*COS(P)+H)
200     Y=INT(V-R*SIN(P)+H)
210     IF P=0
          THEN X1=X:Y1=Y
          ELSE X2=X:Y2=Y:
               LINE (FNX(X1),Y1)-
                    (FNX(X2),Y2):
               X1=X2:Y1=Y2
220 NEXT W
230 A$=INPUT$(1)
240 COLOR 15,4,4 : END
1000 REM subroutine functiewaarde
                    berekenen
1010  R=K*COS(N*P)
1020 RETURN
```

Het programma tekent een achtdelige draaisymmetrische figuur; een bloem met acht blaadjes. Als u met dit programma gaat experimenteren zult u snel ontdekken dat voor even waarden van n een 2n-bladerige en voor oneven waarden van n een n-bladerige bloem ontstaat. Kunt u een tweekleurige bloem maken?

Wilt u dat uw computer sneller tekent? Neem dan een stapgrootte van 3° of van 5° in plaats van 1°. Bedenk dan wel dat de blaadjes wat hoekiger worden.

Als u regel 1010 R=K*COS(N*P) vervangt door één van de onderstaande functies, krijgt u steeds een andere mooie figuur:

```
R = COS(4*SIN(2*P)) : R = ABS(R)
R = COS(4*SIN(3*P)) : R = ABS(R)
R = SIN(3*SIN(2*P)) : R = ABS(R)
R = SIN(5*COS(2*P)) : R = ABS(R)
```

U kunt naar hartelust trigonometrische functies met verschillende parameters 'mengen'. Laat uw creativiteit en nieuwsgierigheid de vrije loop. Probeer eens:

```
1010 R=K*SIN(5*COS(2*SIN(3*COS(4*P)))):R=ABS(R)
```

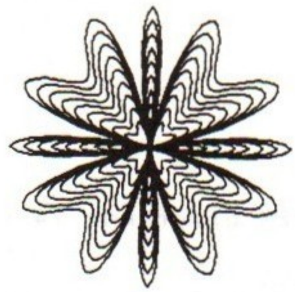
Omdat de poolcoördinaat r per definitie niet negatief mag zijn, moeten we in regel 1010 ook de opdracht R = ABS(R) opnemen.

Nog veel mooiere figuren krijgen we als we niet één grafiek, maar een stelsel van gelijkvormige grafieken tekenen.

**Programma 13** tekent het stelsel krommen:

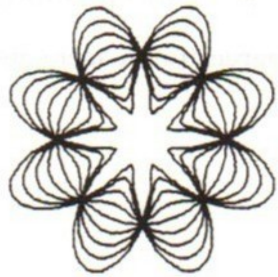
R = K*COS(4*SIN(2*P)).

De parameter K loopt van 20 tot 100 in stappen van 10.



```
100 REM proramma 13 grafiek van de
        functie R=K*cos(4*sin(2*Phi))
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 PI=4*ATN(1)
150 U=128:V=96:H=.5:RD=PI/180
160 FOR K=20 TO 100 STEP 10
170  FOR W=0 TO 360 STEP 2
180   P=W*RD
190   GOSUB 1000 ' R berekenen
200   X=INT(U+R*COS(P)+H)
210   Y=INT(V-R*SIN(P)+H)
220   IF P=0 THEN X1=X:Y1=Y
             ELSE X2=X:Y2=Y:
                  LINE (FNX(X1),Y1)-
                       (FNX(X2),Y2):
                  X1=X2:Y1=Y2
230  NEXT W
240 NEXT K
250 A$=INPUT$(1)
260 COLOR 15,4,4 : END
1000 REM subroutine R berekenen
1010  R=K*COS(4*SIN(2*P))
1020 RETURN
```

Programma 14 tekent sinuskrommen die cirkelvormig gekromd zijn.



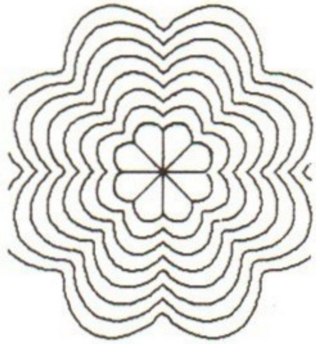
```
100 REM programma 14 sinuskrommen in
                     cirkels
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 PI=4*ATN(1)
150 U=128:V=96:H=.5:RD=PI/180
160 FOR K=-40 TO 40 STEP 10
170  FOR W=0 TO 360 STEP 2
180   P=W*RD
190   GOSUB 1000 ' R berekenen
200   X=INT(U+R*COS(P)+H)
210   Y=INT(V-R*SIN(P)+H)
220   IF P=0 THEN X1=X:Y1=Y
             ELSE X2=X:Y2=Y:
                  LINE(FNX(X1),Y1)-
                      (FNX(X2),Y2):
                  X1=X2:Y1=Y2
230  NEXT W
240 NEXT K
250 A$=INPUT$(1)
260 COLOR 15,4,4 : END
1000 REM subroutine R berekenen
1010  R=60 + K*SIN(4*P)
1020 RETURN
```

Probeert u eens

```
1010 R = 2*TAN(2*P)+K*SIN(2*COS(SIN(6*P)))
```

voor een schitterend spinneweb. De variaties zijn echt onbegrensd!

**Programma 15** tekent een bloemvormige figuur met blaadjes die in het midden aan steeltjes vastzitten. In het programma is N = 4 gekozen. U kunt gerust andere waarden voor N proberen. Het effect is hetzelfde als bij programma 12.



```
100 REM programma 15 bloemen
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 PI=4*ATN(1):N=4:C=.25
150 U=128:V=96:H=.5:RD=PI/180
160 FOR K=30 TO 80 STEP 10
170  FOR W=0 TO 360 STEP 3
180   P=W*RD
190   GOSUB 1000 ' R berekenen
200   X=INT(U+R*COS(P)+H)
210   Y=INT(V-R*SIN(P)+H)
220   IF P=0 THEN X1=X:Y1=Y
             ELSE X2=X:Y2=Y:
                  LINE(FNX(X1),Y1)-
                      (FNX(X2),Y2):
                  X1=X2:Y1=Y2
230  NEXT W
240 NEXT K
250 R=30:P1=180/N*RD
260 FOR J=1 TO N
270    X1=INT(U+R*COS(P)+H)
280    Y1=INT(V-R*SIN(P)+H)
290    X2=INT(U+R*COS(P)+H)
300    Y2=INT(V-R*SIN(P)+H)
310    LINE (FNX(X1),Y1)-(FNX(X2),Y2)
320 NEXT J
330 A$=INPUT$(1)
340 COLOR 15,4,4 : END
1000 REM subroutine R berekenen
1010  R=K*(1+C*ABS(SIN(N*P)))
1020 RETURN
```

U kunt na elke doorloop van de K-lus de afdrukkleur veranderen. Ook de stelen van de bloem kunnen een eigen kleur krijgen. Tekent u een aantal van dergelijke bloemen naast of onder elkaar dan hebt u een begin gemaakt met 'computer-art'.

Spiralen zijn altijd geliefde figuren. Programma 16 tekent logaritmische spiralen of spiralen van Archimedes. Deze laatste soort spiralen hebben als vergelijking

$$r = c.\varphi$$

In het programma op p.40 is voor C de waarde 3 gekozen. Kies gerust andere waarden, maar wel tussen 0,5 en 20.

De logaritmische spiraal heeft (naar Bernoulli) de vergelijking

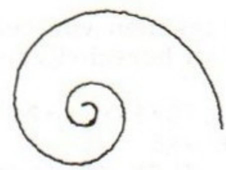
$$r = k\,e^{c\varphi}.$$

In het programma dat de onderste spiraal getekend heeft hebben we voor K de waarde 110 en voor C de waarde -0,2 gekozen.



$r=2\varphi$

$r=110e^{-0,2\varphi}$



Omdat de logaritmische spiraal zich oneindig vaak rond de oorsprong zal winden moet ervoor gezorgd worden dat het programma na bepaalde 'tijd' wordt afgebroken.

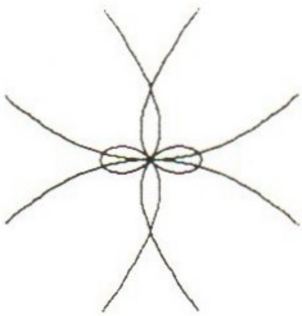
```
100 REM programma 16 spiralen
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 PI=4*ATN(1):C=3
150 U=128:V=96:H=.5:RD=PI/180
160 FOR W=0 TO 10000 STEP 3
170   P=W*RD
180   GOSUB 1000 ' R berekenen
190   X=INT(U+R*COS(P)+H)
200   Y=INT(V-R*SIN(P)+H)
210   IF X<0 OR X>255 OR Y<0 OR Y>191
        THEN A$=INPUT$(1) : STOP
220   IF P=0 THEN X1=X:Y1=Y
            ELSE X2=X:Y2=Y:
                 LINE(FNX(X1),Y1)-
                     (FNX(X2),Y2):
                 X1=X2:Y1=Y2
230 NEXT W
240 A$=INPUT$(1)
250 COLOR 154,4,4 : END
1000 REM subroutine R berekenen
1010  R=C*P
1020 RETURN
```

Voeg toe regel `235 U = 133 : GOTO 160`
en zet in regel 210 achter THEN `GOTO 235` in plaats van STOP
en zie hoe dit de figuur verfraait.

Voor het tekenen van een logaritmische spiraal maken we C gelijk aan -0,2 en herschrijven subroutine 1000 als volgt:

```
1010 R=110*EXP(C*P)
1020 IF R<5
       THEN A$=INPUT$(1) : STOP
1030 RETURN
```

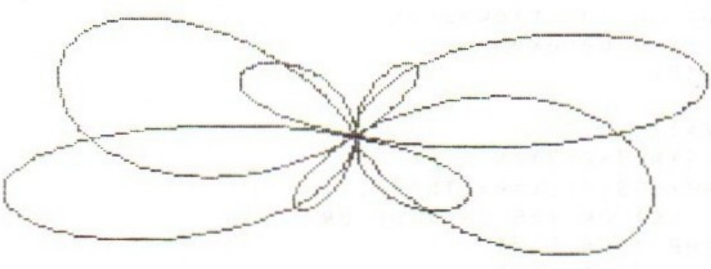
Als laatste programma met poolcoördinaten geven we het **programma 17** voor het tekenen van een willekeurige, in poolcoördinaten geformuleerde, continue of niet-continue functie. De programmastructuur komt overeen met de structuur van programma 11 (zie p.27). U kunt daar de werking van de vlaggen FZ en FA nog eens bestuderen. Ook de andere variabelen hebben dezelfde betekenis als hun naamgenoten in programma 11. Als voorbeeld in het programma hebben we een vrij ingewikkelde functie, die niet overal continu is, gekozen:



$$r = \frac{\sin(1{,}5.\varphi)}{1-2.\cos\varphi}$$

De figuur is getekend met A = -2, B = 2, LP = -2, HP = 2, WO = 0° en WN = 720°.

Wilt u een andere functie proberen, dan hoeft u alleen de regels 1010 en 1020 te veranderen door hierin de nieuwe functie R te berekenen. Mocht er iets fout gaan, dan kan het zijn dat u de volgende regel moet toevoegen: `1025 IF R<0 THEN FZ=1`



Deze zeldzame vlinder heeft als vergelijking $R = \dfrac{4.\sin(1{,}5p+2)}{\cos(p)(1+\dfrac{\cos(3p)}{3})}$ .

Voor A, B, LP, HP, WO en WN nemen we in programma 17:
-4, +4, -4, +4, 0 en 720.

```
100 REM programma 17 grafiek van de
        functie R=f(PHI)
110 CLS
120 INPUT "linkergrens voor X  "; A
130 INPUT "rechtergrens voor X "; B
140 INPUT "ondergrens voor Y   "; LP
150 INPUT "bovengrens voor Y   "; HP
160 INPUT "startwaarde voor PHI"; WO
170 INPUT "eindwaarde voor PHI "; WN
180 PI=4*ATN(1)
190 KX=255/(B-A):KY=191/(HP-LP):H=.5
200 RD=PI/180
210 CLS
220 COLOR 1,15,4 : SCREEN 2
230 DEF FNX(X)=INT(37+X/1.4+.5)
240 FA=1
250 FOR W=WO TO WN
260 P=W*RD
270  X2=INT(KX*(X-A)+H)
280  GOSUB 1000 'functiewaarde
                 berekenen
290  IF FZ=1 THEN FA=1
             ELSE IF FA=1
      THEN X1=X2:FA=0:
       Y1=INT(KY*(HP-Y)+H)
      ELSE Y2=INT(KY*(HP-Y)+H):
       LINE(FNX(X1),Y1)-
       (FNX(X2),Y2):X1=X2:Y1=Y2
300 NEXT W
310 A$=INPUT$(1)
320 COLOR 15,4,4 : END
1000 REM subroutine functiewaarde
                    berekenen
1010  N=1-2*COS(P)
1020  IF N=0
        THEN FZ=1
        ELSE R=SIN(3*P/2)/N
            :X=R*COS(P):Y=R*SIN(P):
             IF X<A OR X>B OR Y<LP OR
             Y>HP THEN FZ=1
                  ELSE FZ=0
1030 RETURN
```

## De parametervorm

De meest interessante krommen krijgen we bij functies waarvan de vergelijking in parametervorm wordt opgesteld. Dit houdt in dat zowel de X- als de Y-coördinaat als functie van dezelfde, derde, parameter t worden uitgedrukt. In de natuurkunde zien we vaak de tijd als parameter (vandaar de t). In de wiskunde is de parameter t bijna altijd een hoek in radialen.

Zo is de parametervorm van een cirkel met straal r en het middelpunt in de oorsprong:

$$x = r\cos t \qquad \text{en} \qquad y = r\sin t$$

Uit deze twee vergelijkingen kan gemakkelijk de cartesische vorm $x^2+y^2=r^2$ gedistilleerd worden. De vergelijking van een ellips met het middelpunt in de oorsprong en met een halve lange as a en een halve korte as b is:

$$x = a\cos t \qquad \text{en} \qquad y = b\sin t$$

Het is eenvoudig programma's te ontwikkelen voor het tekenen van stelsels concentrische of excentrische cirkels. Ook stelsels ellipsen zijn, dankzij de parametervorm, eenvoudig te tekenen. We doen dit echter niet, omdat de figuren niet zo bijzonder zijn en vrij bekend. In hoofdstuk 1 hebben we trouwens al een ellips met behulp van de parametervorm geprogrammeerd.

De volgende programma's tekenen figuren die u mogelijk nog niet kent en die zich voor computer-graphics bijzonder goed lenen.

**Programma 18** tekent een Lissajousfiguur. In het algemeen is de parametervorm voor een dergelijke figuur:

$$x = k_1.\sin(f_1.t+p_1) + k_2.\cos(f_2.t)$$
$$y = k_3.\sin(f_3.t+p_3) + k_4.\cos(f_4.t)$$

Hierin is p de fase, f de frequentie en k de amplitude. Om een mooie figuur te krijgen hebben we de volgende waarden gekozen:

$$k_1=k_3=100,\ f_1=16,\ p_1=0,\ f_3=17,\ p_3=35 \text{ en } k_2=k_4=f_2=f_4=0$$

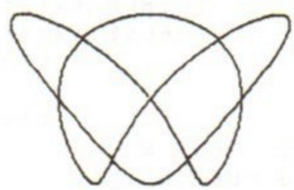
```
100 REM programma 18 lissajousfiguren
110 CLS
120 PRINT "toets K1,F1,P1,K2,F2 in";
130 INPUT K1,F1,P1,K2,F2
140 PRINT:PRINT
150 PRINT "toets K3,F3,P3,K4,F4 in";
160 INPUT K3,F3,P3,K4,F4
170 PI=4*ATN(1)
180 U=128:V=96:H=.5:RD=PI/180
190 CLS
200 COLOR 1,15,4 : SCREEN 2
210 DEF FNX(X)=INT(37+X/1.4+.5)
220 FOR W=0 TO 360
230     T=W*RD
240     GOSUB 1000 'x en y berekenen
250     X=INT(U+X+H)
260     Y=INT(V-Y+H)
270     IF W=0
          THEN X1=X:Y1=Y
          ELSE X2=X:Y2=Y:
               LINE(FNX(X1),Y1)-
                   (FNX(X2),Y2):
               X1=X2:Y1=Y2
280 NEXT W
290 A$=INPUT$(1)
300 COLOR 15,4,4 : END
1000 REM subroutine x en y berekenen
1010  X=K1*SIN(F1*T+P1)+K2*COS(F2*T)
1020  Y=K3*SIN(F3*T+P3)+K4*COS(F4*T)
1030 RETURN
```



Experimenteer met het programma! Zoek zelf de juiste waarden voor de constanten zodat u mooie figuren krijgt. Natuurkundigen zullen u hierbij graag helpen; zij weten welke waarden u moet nemen!

**Programma 19** presenteert een niet alledaagse figuur. Deze figuur wordt dikwijls de 'vliegekopkromme' genoemd. Om de vliegekop horizontaal op het beeldscherm te krijgen moet de parameter t het interval 90°-450° doorlopen.

```
100 REM programma 19 vliegekopfiguur
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 PI=4*ATN(1)
150 U=128:V=96:K=30:H=.5:RD=PI/180
160 FOR W=90 TO 450 STEP 3
170    T=W*RD
180    GOSUB 1000 'x en y berekenen
190    X=INT(U+X+H)
200    Y=INT(V-Y+H)
210    IF W=90
         THEN X1=X:Y1=Y
         ELSE X2=X:Y2=Y:
              LINE(FNX(X1),Y1)-
                  (FNX(X2),Y2):
              X1=X2:Y1=Y2
220 NEXT W
230 A$=INPUT$(1)
240 COLOR 15,4,4 : END
1000 REM subroutine x en y berekenen
1010  X=K*SIN(2*T)*(2.5+COS(3*T))
1020  Y=K*2*COS(3*T)
1030 RETURN
```



Het tekenen van de x- en y-as is achterwege gelaten om de 'kop' beter te laten uitkomen.

Voor een echte vliegekop met 'ogen' moet u in de appendix kijken.

Hoe geraffineerder u de functies x = f(t) en y = f(t) kiest, des te ongewoner worden de figuren. Probeer hier hoe creatief u kunt zijn.

De 'huiswiskundigen' bij computerfabrikanten hebben vaak hun eigen lievelingsfuncties. Die zie je dan ook vaak in een demonstratieprogramma optreden.

Programma 20 tekent een stelsel krommen. Hewlett-Packard publiceerde deze tekening enige tijd geleden. Onder het programma staat een tabel met waarden voor A en B, die heel mooie figuren opleveren. Probeer zelf andere combinaties. U zult verrukt zijn over wat u ziet!



A = -6 / B = 1

toets in: -6,1

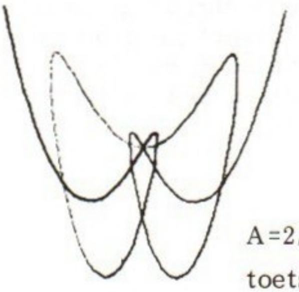
```
100 REM programma 20 vlinders
110 CLS
120 PI=4*ATN(1)
130 PI=4*ATN(1)
140 U=128:V=92:H=.5:RD=PI/180
150 KX=10:KY=10
160 INPUT "toets A en B in"; A,B
170 CLS
180 COLOR 1,15,4 : SCREEN 2
190 DEF FNX(X)=INT(37+X/1.4+.5)
200 FOR N=-3 TO 3
210  FOR W=0 TO 360 STEP 3
220   T=W*RD
230   X=(A+B)*COS(T)-
               N*B*COS((A+B)/B*T)
240   Y=(A+B)*SIN(T)-
               N*B*SIN((A+B)/B*T)
250    XX=INT(U+KX*X+H)
260    YY=INT(V-KY*Y+H)
270    IF W=0
         THEN X1=XX:Y1=YY
         ELSE X2=XX:Y2=YY:
                LINE(FNX(X1),Y1)-
                    (FNX(X2),Y2):
                X1=X2:Y1=Y2
280  NEXT W
290 NEXT N
300 A$=INPUT$(1)
310 COLOR 15,4,4 : END
```

Tabel voor mooie figuren

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | -6 | -6 | -8 | 4 | 4 | 6 | 4,5 | ....... |
| B | 1 | 2 | 2 | 1 | 2 | 1 | 1,5 | ....... |

Kies in deze programma's de stapgrootte voor hoek W (FOR W=.....) gerust 1; dit duurt wat langer maar geeft mooiere tekeningen.

Het laatste programma uit dit hoofdstuk tekent 'supersymmetrische' figuren. Ook nu geven we een tabel met waarden voor de parameters A, B en C. Het is ongelooflijk hoe de figuur totaal verandert als we andere waarden voor één of meer parameters kiezen. Het idee voor programma 21 is afkomstig uit het Amerikaanse tijdschrift Creative Computing. Het kan soms even duren voor het tekenen wordt hervat. Druk niet direct op een toets, want dan wordt na het tekenen direct teruggeschakeld naar het gewone scherm en bent u uw tekening kwijt. Druk pas op een toets als u zeker weet dat de tekening af is.

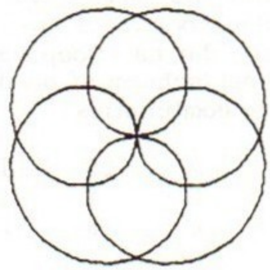


A=2/B=7/C=3

toets in: 2,7,3

A=6/B=6/C=4

toets in: 6,6,4



```
100 REM programma 21 symmetrische
                     figuren
110 CLS
120 PI=4*ATN(1)
130 U=128:V=92:H=.5:RD=PI/180:K=100
140 INPUT "toets A,B en C in"; A,B,C
150 CLS
160 COLOR 1,15,4 : SCREEN 2
170 DEF FNX(X)=INT(37+X/1.4+.5)
180 FOR W=0 TO 360
190   T=W*RD:R=K*SIN(C*T)
200   X2=INT(U+R*COS(A*T)+H)
210   Y2=INT(V-R*SIN(B*T)+H)
220    IF W=0
         THEN X1=X2:Y1=Y2
         ELSE LINE(FNX(X1),Y1)-
                   (FNX(X2),Y2):
              X1=X2:Y1=Y2
230 NEXT W
240 A$=INPUT$(1)
250 COLOR 15,4,4 : END
```

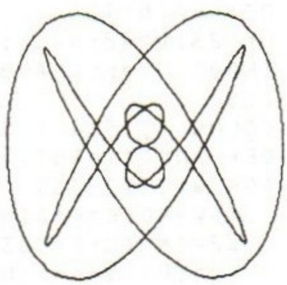
Kijk in de appendix hoe u de waarden van A, B en C links naast de figuur op het graphic-scherm kunt afdrukken.

Tabel voor mooie figuren

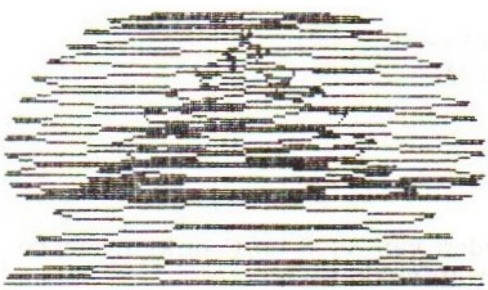
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | 2 | 6 | 4 | 1 | 3 | 2 | ............... |
| B | 7 | 6 | 6 | 1 | 3 | 2 | ............... |
| C | 3 | 4 | 1 | 4 | 5 | 9 | ............... |

Het zou jammer zijn als u bij het zien van een mooie figuur de waarden voor A, B en C niet hebt genoteerd. Verander het programma zo, dat na afloop van het tekenen de waarden van A, B en C op het beeldscherm of op de printer worden afgedrukt. Ook heel leuk zijn de combinaties:

| | | | |
|---|---|---|---|
| A | 20 | -40 | 100 |
| B | -1 | -40 | -0,5 |
| C | 3 | 10 | 3 |



A=4/B=6/C=1



A=-1/B=-40/C=0,5

# 4 Tekenen van driedimensionale figuren

In dit en het volgende hoofdstuk gaan we ons bezighouden met het tekenen van driedimensionale lichamen (zo heet dat in de wiskunde) zoals kubussen, prisma's, pyramiden, kegels en bollen, en met het tekenen van driedimensionale grafieken van functies. Zo'n driedimensionale grafiek is een vlak in de ruimte met een vergelijking van de vorm $z = f(x,y)$. Zo krijgen we de bekende 'Mexicaanse hoed' (zie p.73) als we de functie $z = e^{-(x^2+y^2)}$ tekenen (e is het grondtal van de natuurlijke logaritme; e = 2,71828...).

Er bestaan veel programma's waarmee driedimensionale figuren getekend kunnen worden. Waarom we toch ook in dit boek dergelijke programma's laten zien komt, omdat er aan de meeste van deze programma's drie nadelen kleven:

1. De meeste programma's voor driedimensionaal-tekenen zijn voor een bepaald graphics-systeem ontworpen en zijn slechts met veel inspanning geschikt te maken voor andere systemen.

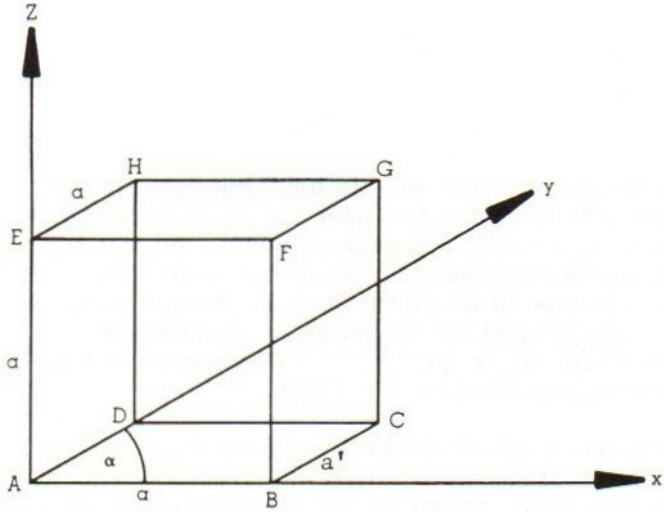
2. De wiskundige theorie die aan het driedimensionaal-tekenen ten grondslag ligt wordt zelden of zeer summier behandeld.

3. Veel programma's zijn uiterst ingenieus ontworpen en draaien op de kleine microcomputers, in BASIC, heel langzaam.

Met de volgende programma's en stukjes theorie hopen we deze drie nadelen uit de weg te ruimen.

Er bestaan verschillende methoden om een driedimensionaal lichaam, bijvoorbeeld een kubus, in een plat vlak te projecteren. Wij hanteren de projectiemethode, die u wellicht op school gebruikt hebt (of nog gebruikt). Stelt u zich een doorzichtige kubus voor met ribben van draadijzer. U staat voor deze kubus een beetje rechts van het

midden en kijkt er zo'n beetje schuin bovenop. Uw gezichtsstralen projecteren elke hoek, met de daaraan verbonden ribben, van de kubus in een, achter de kubus liggend, projectievlak. Zo ontstaat de bekende 'schuine' afbeelding van een kubus.




De verticale en horizontale ribben worden op ware grootte getekend. De ribben die naar achteren lopen worden verkort weergegeven.

a' = k.a    k = verkleiningsfactor

De rechte hoek tussen de ribben BA en AD in het grondvlak lijkt eveneens kleiner te zijn geworden (zie α in de bovenstaande figuur). U kunt gemakkelijk nagaan dat door het vastleggen van alpha (bijvoorbeeld 45°) en k (bijvoorbeeld 0,5) de projectierichting eenduidig bepaald is.

Wat we nodig hebben zijn transformatievergelijkingen die voor een bepaalde α en k de coördinaten x,y,z van een punt op de ruimtelijke kubus projecteren op de coördinaten x' en y' van het geprojecteerde punt in het platte (projectie-) vlak. Deze formules vormen de kern van alle volgende programma's.

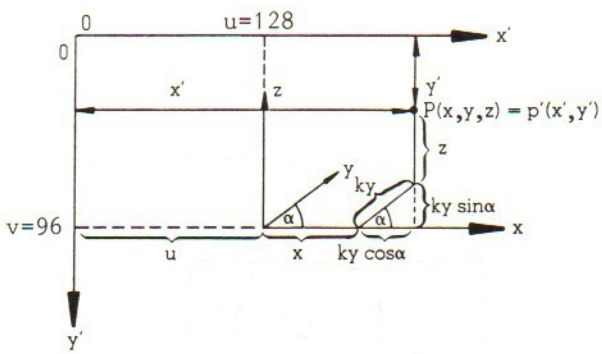
## Afleiding van de transformatievergelijkingen

Het projectievlak is ons beeldscherm. We gebruiken de MSX-schermresolutie van 256 bij 192 punten. De oorsprong van het ruimtelijke coördinatensysteem (xyz) moet precies in het midden van het beeldscherm geprojecteerd worden. De oorsprong van het beeldschermcoördinatenstelsel ligt weer in de linkerbovenhoek. De x-as wijst naar rechts; de y-as naar beneden.

De afbeelding van het ruimtelijke punt P(x,y,z) op het punt P'(x',y') in het platte vlak komt tot stand met behulp van de volgende transformatievergelijkingen:

$x' = U + x + k.y.\cos\alpha$ èn
$y' = V - (k.y.\sin\alpha + z)$

Hoe we aan deze vergelijkingen komen is uit onderstaande figuur direct (nou ja, direct.....) duidelijk.



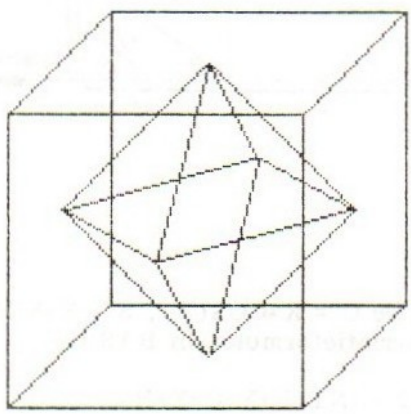

Nemen we C = K*COS(A), S = K*SIN(A) en H = 0,5, dan zijn de transformatieformules in BASIC:

```
X2 = INT(U+X+C*Y+H)
Y2 = INT(V-S*Y-Z+H)
```

Meer wiskunde hebben we voor dit hoofdstuk niet nodig.

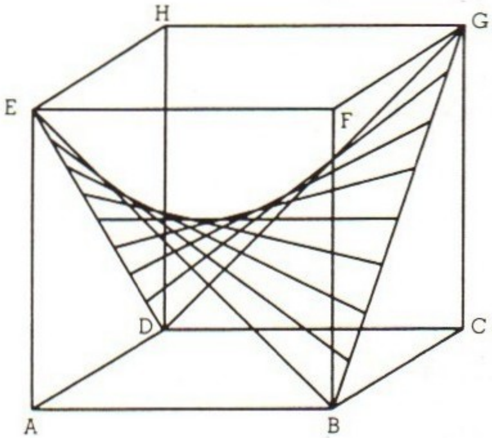
Programma 22 laat zien hoe je van elk, door rechte lijnen begrensd lichaam (polyeder) een projectie kan maken. Hiervoor is het noodzakelijk dat de waarden van de coördinaten x,y,z in alle hoekpunten bekend zijn. Als voorbeeld van het gebruik van dit programma tekenen we een kubus met een ingeschreven achtvlak (oktaëder). De gevolgde programmeertechniek leidt ook bij een viervlak, prisma of pyramide tot het gewenste resultaat.

We nemen aan dat het middelpunt van de kubus samenvalt met de oorsprong van het ruimtelijke coördinatensysteem. Om het programma 'sneller' te maken nemen we de coördinaten van de acht hoekpunten ABCDEFGH van de kubus en de coördinaten van de zes hoekpunten IJKLMN van de achthoek op in DATAregels. Het tekenen van de ribben wordt door de letterstring Z$ bestuurd. Zo betekent Z$ = "ABBCCDDA" dat de computer van A naar B, van B naar C, van C naar D en van D naar A rechte lijnen moet trekken. Met de MID$-functie halen we steeds één letter uit de string Z$ en maken er vervolgens met de ASC-functie een getal van (A=1, B=2, C=3, .....). Hiermee moet de werking van het programma duidelijk zijn. Kies voor alpha 45° en voor k 0,5; dat geeft de mooiste projectie.

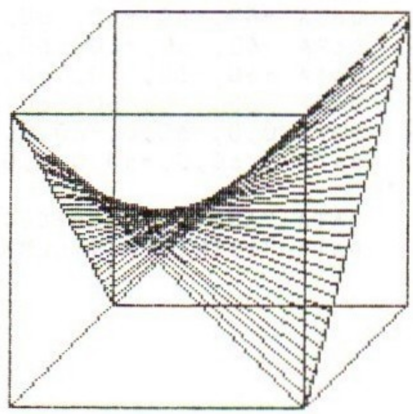


```
100 REM programma 22 kubus
                     met achtvlak
110 CLS
120 INPUT "alfa in graden (45)     ";A
130 INPUT "verkleiningsfactor (.5)";K
140 PI=4*ATN(1)
150 U=128:V=96:H=.5:RD=PI/180
160 W=A*RD : C=K*COS(W) : S=K*SIN(W)
170 DIM X(14),Y(14),Z$(2)
180 FOR J=1 TO 14
190    READ X,Y,Z
200    X(J)=INT(U+X+C*Y+H)
210    Y(J)=INT(V-S*Y-Z+H)
220 NEXT J
230 CLS
240 COLOR 1,15,4 : SCREEN 2
250 DEF FNX(X)=INT(37+X/1.4+.5)
260 FOR N=1 TO 2
270  READ Z$(N) : L=LEN(Z$(N))
280  FOR M=1 TO L-1 STEP 2
290   I=ASC(MID$(Z$(N),M,1))-64
300   J=ASC(MID$(Z$(N),M+1,1))-64
310   LINE (FNX(X(I)),Y(I))-
           (FNX(X(J)),Y(J))
320  NEXT M
330 NEXT N
340 A$=INPUT$(1)
350 COLOR 15,4,4 : END
360 '
370   DATA -60,-60,-60, 60,-60,-60
380   DATA  60, 60,-60,-60, 60,-60
390   DATA -60,-60, 60, 60,-60, 60
400   DATA  60, 60, 60,-60, 60, 60
410   DATA 0,0,-60,0,-60,0,60,0,0
420   DATA 0,60,0,-60,0,0,0,0,60
430 '
440   DATA "ABBCCDDAAEBFCGDHEFFGGHHE"
450   DATA "IJIKILIMJKKLLMMJJNKNLNMN"
```

In programma 23 wordt in een kubus een zadelvlak getekend. Het geeft het idee van een gekromd vlak in een kubus.




Bekijk de bovenstaande figuur. De diagonalen BG en ED zijn elk in acht (n) gelijke stukken verdeeld en de zo ontstane punten zijn paarsgewijs door een rechte lijn met elkaar verbonden. Kies in het programma voor n de waarde 18 of 15.



```
100 REM programma 23 kubus
                      met zadelvlak
110 CLS
120 INPUT "alfa in graden (45)      ";A
130 INPUT "verkleiningsfactor (.5)";K
140 INPUT "hoeveel lijnen (32)      ";N
150 PI=4*ATN(1)
160 U=128:V=96:H=.5:RD=PI/180
170 W=A*RD : C=K*COS(W) : S=K*SIN(W)
180 DIM X(8),Y(8)
190 FOR J=1 TO 8
200    READ X,Y,Z
210    X(J)=INT(U+X+C*Y+H)
220    Y(J)=INT(V-S*Y-Z+H)
230 NEXT J
240 CLS
250 COLOR 1,15,4 : SCREEN 2
260 DEF FNX(X)=INT(37+X/1.4+.5)
270 READ Z$ : L=LEN(Z$)
280 FOR M=1 TO L-1 STEP 2
290   I=ASC(MID$(Z$,M,1))-64
300   J=ASC(MID$(Z$,M+1,1))-64
310   LINE (FNX(X(I)),Y(I))-
           (FNX(X(J)),Y(J))
320 NEXT M
330 FOR J=0 TO N
340   X1=INT(X(2)+J*(X(7)-X(2))/N+H)
350   Y1=INT(Y(2)+J*(Y(7)-Y(2))/N+H)
360   X2=INT(X(5)+J*(X(4)-X(5))/N+H)
370   Y2=INT(Y(5)+J*(Y(4)-Y(5))/N+H)
380   LINE(FNX(X1),Y1)-(FNX(X2),Y2)
390 NEXT J
400 LINE (FNX(X(2)),Y(2))-
         (FNX(X(7)),Y(7))
410 LINE (FNX(X(5)),Y(5))-
         (FNX(X(4)),Y(4))
420 A$=INPUT$(1)
430 COLOR 15,4,4 : END
440 '
450   DATA -60,-60,-60, 60,-60,-60
460   DATA  60, 60,-60,-60, 60,-60
470   DATA -60,-60, 60, 60,-60, 60
480   DATA  60, 60, 60,-60, 60, 60
490   DATA "ABBCCDDAAEBFCGDHEFFGGHHE"
```

Met programma 24 kunnen we, naar keuze, cilinders, kegels of afgeknotte kegels tekenen. Als we voor R1 en R2 dezelfde waarde invoeren, krijgen we een cilinder. Als R2 kleiner is dan R1 krijgen we een afgeknotte kegel en voor R2 = 0 krijgen we een kegel.

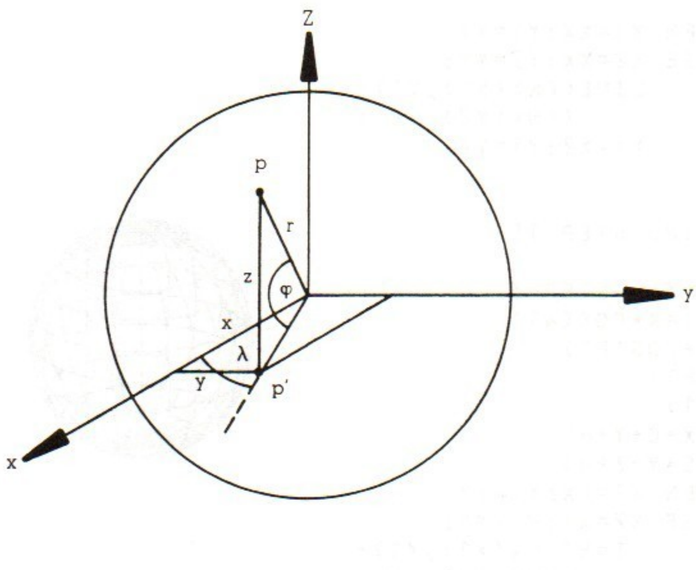
We plaatsen het lichaam zo dat het grondvlak bij z = -60 ligt en het bovenvlak (de top) bij z = +60. Het programma tekent zeven breedtecirkels met een onderlinge afstand van z = 20 en 16 lijnen op de kegel- (of cilinder-) mantel loodrecht op de breedtecirkels. Tot slot wordt verticaal de z-as getekend.

```
100 REM programma 24 cylinders en
                         kegels
110 CLS
120 INPUT "alfa in graden (45)    ";A
130 INPUT "verkleiningsfactor(.5)";K
140 INPUT "stralen R1 en R2  ";R1,R2
150 PI=4*ATN(1)
160 U=128:V=96:H=.5:RD=PI/180
170 DR=(R1-R2)/6:N=0
180 W=A*RD : C=K*COS(W) : S=K*SIN(W)
190 CLS
200 COLOR 1,15,4 : SCREEN 2
210 DEF FNX(X)=INT(37+X/1.4+.5)
220 FOR Z=-60 TO 60 STEP 20
230  R=R1-N*DR
240  FOR W=0 TO 360 STEP 3
250   W1=W*RD
260   X=R*COS(W1):Y=R*SIN(W1)
270   XX=INT(U+X+C*Y+H)
280   YY=INT(V-S*Y-Z+H)
290   IF W=0 THEN X1=XX:Y1=YY
             ELSE X2=XX:Y2=YY:
                  LINE(FNX(X1),Y1)-
                      (FNX(X2),Y2):
                  X1=X2:Y1=Y2
300  NEXT W
310  N=N+1
320 NEXT Z
330 FOR W=0 TO 360 STEP 23
340    W1=W*RD
350    X=R1*COS(W1):Y=R1*SIN(W1)
360    X1=INT(U+X+C*Y+H)
370    Y1=INT(V-S*Y+60+H)
380    X=R2*COS(W1):Y=R2*SIN(W1)
390    X2=INT(U+X+C*Y+H)
400    Y2=INT(V-S*Y-60+H)
410    LINE(FNX(X1),Y1)-(FNX(X2),Y2)
420 NEXT W
430 LINE (FNX(U),0)-(FNX(U),192)
440 A$=INPUT$(1)
450 COLOR 15,4,4 : END
```



Erg leuk is het tekenen van een bol met breedtelijnen en meridianen. Zoals bekend is kunnen we elk punt op het boloppervlak eenduidig vastleggen met de straal van de bol (r), de geografische breedte (φ) en de geografische lengte (λ) (zie figuur). Het omzetten van deze bolcoördinaten r,φ,λ in cartesische coördinaten (x,y,z) geschiedt met de volgende drie vergelijkingen:

$x = r\cos\varphi\cos\lambda$
$y = r\cos\varphi\sin\lambda$
$z = r\sin\varphi$




De bolvergelijking in cartesische vorm is overigens $x^2+y^2+z^2=r^2$.

In **programma 25** wordt de hoek φ door de W en de hoek λ door P vertegenwoordigd. Kies voor alpha 90° en voor k 0,5. Andere waarden vervormen de bol. Zie de appendix voor mooiere bollen.

```
100 REM programma 25 bol
110 CLS
120 INPUT "alfa in graden (90)      ";A
130 INPUT "verkleiningsfactor(.5)";K
140 INPUT "straal,maximaal 75       ";R
150 PI=4*ATN(1)
160 U=128:V=96:H=.5:RD=PI/180
170 W=A*RD : C=K*COS(W) : S=K*SIN(W)
180 CLS
190 COLOR 1,15,4 : SCREEN 2
200 DEF FNX(X)=INT(37+X/1.4+.5)
210 FOR W=-90 TO 90 STEP 15
220  W1=W*RD:R1=R*COS(W1)
230  FOR P=0 TO 360 STEP 3
240   P1=P*RD
250   X=1.15*R1*COS(P1):Y=R1*SIN(P1):
      Z=R*SIN(W1)
260   XX=INT(U+X+SGN(P)*C*Y+H)
270   YY=INT(V-SGN(P)*S*Y-Z+H)
280   IF P = 0
             THEN X1=XX:Y1=YY
             ELSE X2=XX:Y2=YY:
                  LINE(FNX(X1),Y1)-
                      (FNX(X2),Y2):
                  X1=X2:Y1=Y2
290  NEXT P
300 NEXT W
310 FOR P=0 TO 180 STEP 15
320  P1=P*RD
330  FOR W=0 TO 360 STEP 3
340   W1=W*RD:R1=R*COS(W1)
350   X=1.15*R1*COS(P1)
360   Y=R1*SIN(P1)
370   Z=R*SIN(W1)
380   XX=INT(U+X+C*Y+H)
390   YY=INT(V-S*Y-Z+H)
400   IF W=0 THEN X1=XX:Y1=YY
             ELSE X2=XX:Y2=YY:
                  LINE(FNX(X1),Y1)-
                      (FNX(X2),Y2):
                  X1=X2:Y1=Y2
410  NEXT W
420 NEXT P
430 A$=INPUT$(1)
440 COLOR 15,4,4 : END
```



Een geliefd demonstratieprogramma van computerleveranciers is een om zijn as draaiend driedimensionaal lichaam. Meestal wordt als 'draai-as' de z-as gekozen, die dan samenvalt met de lengte-as van het lichaam.

Microcomputers die alleen een BASIC-vertolker bezitten zijn te langzaam om zo'n draaiing real-time uit te voeren. Er zijn namelijk nogal ingewikkelde trigonometrische berekeningen voor nodig, die tamelijk veel tijd in beslag nemen. Daarnaast duurt het tekenen van het lichaam in een bepaalde stand in BASIC zo'n 1 à 2 seconden. Zouden we in een BASIC-programma steeds het lichaam tekenen, uitwissen, draaien, tekenen, uitwissen, draaien, ... enz., dan zien we het draaiende lichaam in plaats van een vloeiende beweging een schokkerige beweging maken. De beste oplossing voor dit probleem is de volgende.

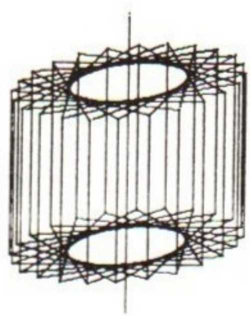
Deel de totale middelpuntshoek van 360° in n even grote hoeken. Voor elk van deze hoeken

$$\omega_1 = \frac{360°}{n}, \quad \omega_2 = 2.\omega_1, \; \ldots\ldots, \; \omega_n = n.\omega_1 = 360°$$

berekenen we van te voren de coördinaten van de hoekpunten van het lichaam. Voor elke stand slaan we deze hoekpuntscoördinaten in een array op. Al deze berekeningen voeren we in BASIC uit bij een leeg beeldscherm. Als de berekeningen klaar zijn wordt een machinetaalprogramma uitgevoerd dat de eerste groep coördinaten uit de array haalt, het lichaam tekent, na een bepaalde tijd het beeldscherm wist, de volgende groep coördinaten ophaalt, ..... enz. Deze oplossing heeft een redelijk vloeiende draaiing tot gevolg. Dit programma is geschreven op een CBM 3016 die een 6502-microprocessor bezit. Hier doen we echter anders (zie volgende pagina).

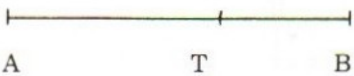
Programma 26 laat een driezijdig prisma om de z-as draaien. De z-as is zowel de lengte-as als de symmetrie-as van het prisma. Het programma tekent een prisma en wacht vervolgens met het draaien en opnieuw tekenen tot we een toets indrukken. Elk volgend prisma wordt over zijn voorgangers heen getekend. We kunnen het draaien dus vertraagd gadeslaan.

```
100 REM programma 26 draaiend prisma
110 CLS
120 INPUT "alfa in graden (45)    ";A
130 INPUT "verkleiningsfactor(.5)";K
140 INPUT "draaihoek in gr. (45)";OM
150 PI=4*ATN(1)
160 U=128:V=96:H=.5:RD=PI/180:R=80
170 W=A*RD : C=K*COS(W) : S=K*SIN(W)
180 CLS
190 COLOR 1,15,4 : SCREEN 2
200 DEF FNX(X)=INT(37+X/1.4+.5)
210 LINE (FNX(U),0)-(FNX(U),192)
220 DIM X(7),Y(7)
230 FOR W=0 TO 360 STEP OM
240  FOR J=0 TO 3
250   W1=(W+J*120)*RD
260   X=R*COS(W1)
270   Y=R*SIN(W1)
280   X(J)=INT(U+X+C*Y+H)
290   Y(J)=INT(V-S*Y+60+H)
300   X(J+4)=X(J)
310   Y(J+4)=INT(V-S*Y-60+H)
320  NEXT J
330 '
340  FOR J=0 TO 2
350      LINE (FNX(X(J)),Y(J))-
              (FNX(X(J+1)),Y(J+1))
360      LINE (FNX(X(J)),Y(J))-
              (FNX(X(J+4)),Y(J+4))
370      LINE (FNX(X(J+4)),Y(J+4))-
              (FNX(X(J+5)),Y(J+5))
380  NEXT J
390  A$=INPUT$(1)
400 NEXT W
410 COLOR 15,4,4 : END
```



Programma 27 tekent de projectie van een regelmatig twintigvlak (ikosaëder). Het is bekend dat er slechts vijf 'echt-regelmatige' veelvlakken (polyeders) zijn, namelijk een tetraëder (regelmatig drievlak), een kubus (regelmatig zesvlak), een octaëder (regelmatig achtvlak), een dodekaëder (regelmatig twaalfvlak) en een ikosaëder (regelmatig twintigvlak). Een ikosaëder heeft 12 hoeken, 30 ribben en 20 vlakken. Een vlak is een gelijkzijdige driehoek. Als we programma 22 willen gebruiken om zo'n ikosaëder te tekenen dan hebben we de coördinaten van alle 12 hoekpunten nodig. Als we het ikosaëder in een speciale stand zetten kunnen we de hoekpuntcoördinaten met behulp van de techniek van de 'gulden snede' relatief eenvoudig berekenen. Wie zich voor de hierachterliggende wiskundige theorie interesseert moet er maar eens een meetkundeboek op naslaan. De 'gulden-snede'-techniek deelt een lijnstuk AB in twee stukken AT en TB op zo'n manier dat de volgende evenredigheid geldt:

$$AB : AT = AT : TB$$




Kiezen we voor AB de lengte 1, dan kunnen we de lengte van AT berekenen; immers

$$\frac{AB}{AT} = \frac{AT}{TB}, \quad \text{dus} \quad \frac{1}{t} = \frac{t}{1-t}, \quad \text{als } t = AT$$

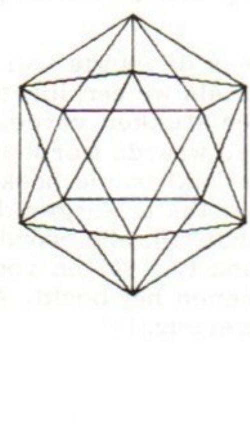
dus dan moet gelden $t^2 = 1-t$ ofwel $t^2+t-1=0$.
Met de alombekende abc-formule berekenen we nu

$$t_{1,2} = \frac{-1 \pm \sqrt{1+4}}{2}$$

De positieve wortel geeft: $t = \frac{-1+\sqrt{5}}{2}$

Dit is de lengte van het grootste van de twee stukken die we krijgen als we een lijnstuk, ter lengte 1, volgens de 'gulden snede' in twee stukken verdelen. ($t = 0,618$; $1-t = 0,382$).
Deze waarde wordt in regel 170 berekend en in de regels 220-330 gebruikt om de hoekpuntcoördinaten van het ikosaëder te berekenen. Om te zorgen dat het twintigvlak voldoende groot getekend wordt zijn alle coördinaten met 80 (F) vermenigvuldigd. Kies voor alpha (A) 90° en voor k (K) de waarde 0,4. Andere waarden vertekenen het beeld. Alpha = 180° en K=0,6 geeft trouwens iets onverwachts!

```
100 REM programma 27 ikosaeder
110 CLS
120 INPUT "alfa in graden (90)  ";A
130 INPUT "verkleiningsfactor(.4)";K
140 PI=4*ATN(1)
150 U=128:V=96:H=.5:RD=PI/180
160 W=A*RD : C=K*COS(W) : S=K*SIN(W)
170 T=(SQR(5)-1)/2 : F=80
180 CLS
190 COLOR 1,15,4 : SCREEN 2
200 DEF FNX(X)=INT(37+X/1.4+.5)
210 DIM X(12),Y(12),Z(12),Z$(3)
220 X(1)=    0: Y(1)=  F*T: Z(1)=  -F
230 X(2)=  0: Y(2)= -F*T: Z(2)=  -F
240 X(3)=    F: Y(3)=    0: Z(3)=-F*T
250 X(4)=   -F: Y(4)=    0: Z(4)=-F*T
260 X(5)= F*T: Y(5)=     F: Z(5)=   0
270 X(6)=-F*T: Y(6)=     F: Z(6)=   0
275 X(7)= F*T: Y(7)=    -F: Z(7)=   0
280 X(8)=-F*T: Y(8)=    -F: Z(8)=   0
290 X(9)=    F: Y(9)=    0: Z(9)= F*T
300 X(10)= -F: Y(10)=   0:Z(10)= F*T
310 X(11)=  0: Y(11)= F*T:Z(11)=   F
320 X(12)=  0: Y(12)=-F*T:Z(12)=   F
330 FOR N=1 TO 3
340  READ Z$(N) : L=LEN(Z$(N))
350  FOR M=1 TO L-1 STEP 2
360   I=ASC(MID$(Z$(N),M,1))-64
370   J=ASC(MID$(Z$(N),M+1,1))-64
380   X1=INT(U+X(I)+C*Y(I)+H)
390   Y1=INT(V-S*Y(I)-Z(I)+H)
400   X2=INT(U+X(J)+C*Y(J)+H)
410   Y2=INT(V-S*Y(J)-Z(J)+H)
420   LINE (FNX(X1),Y1)-(FNX(X2),Y2)
430  NEXT M
440 NEXT N
450 A$=INPUT$(1)
460 COLOR 15,4,4 : END
470 '
480   DATA "BCCEEFFDDBABACAEAFAD"
490   DATA "BHHDDJJFFKKEEIICCGGB"
500   DATA "GHHJJKKIIGLGLHLJLKLI"
```



# 5 Het tekenen van vlakken in de ruimte

In het vorige hoofdstuk hebben we ons uitvoerig beziggehouden met het tekenen van driedimensionale lichamen. We herhalen hiervan nog eens de belangrijkste punten:

1. We gebruiken de parallelprojectie om een driedimensionaal lichaam met n hoekpunten P(x,y,z) in een plat vlak te tekenen. Voor de projectiehoek alpha (α) gebruiken we bij voorkeur de waarde 45° of 60°. De schuin-naar-achteren lopende ribben worden korter getekend dan ze in werkelijkheid zijn. Als verkleiningsfactor kiezen we vaak 1/2 of 1/3.

2. Elk punt P(x,y,z) van het ruimtelijke lichaam wordt op een punt P'(x',y') in het projectievlak geprojecteerd. De hierbij gebruikte projectieformules zijn:

   $$x' = U + x + k.y\text{-}\cos\alpha$$
   $$y' = V - k.y.\sin\alpha - z$$

   Zoals gebruikelijk zijn U en V de coördinaten van het midden van het beeldscherm. Tot nu toe hebben we U = 128 en V = 96 verondersteld.

In de volgende BASIC-programma's zien we de bovenstaande transformatievergelijkingen in de vorm:

```
XG = INT(U+XX+C*YY+H)
YG = INT(V-S*YY-Z+H)
```

De betekenis van de gebruikte variabelen is:

| | | |
|---|---|---|
| XG,YG | : | beeldschermcoördinaten x',y' |
| XX,YY | : | de lopende coördinaten x,y |
| C | : | K*COS(W*RD) |
| S | : | K*SIN(W*RD) |
| W | : | projectiehoek in graden |

RD : factor $\pi/180$ voor omrekenen van graden in radialen
K : verkleiningsfactor

### Tekenen van driedimensionale functies

Als paradepaardje van menige demonstratie van Hoge-Resolutie-Graphics wordt vaak een programma gebruikt dat een of andere fraaie grafiek van een driedimensionale functie tekent. Als leek vraag je je af hoe ze weten welke functies ze moeten nemen, hoe het tekenen van de grafiek van zo'n functie geprogrammeerd wordt, en hoe ze het programmatechnisch voor elkaar krijgen dat de 'onzichtbare lijnen' ook inderdaad niet getekend worden. Bij dergelijke demonstraties wordt doorgaans geen programmalisting getoond. Mocht dit wel het geval zijn dan is het programma vaak zo machine-afhankelijk dat het omzetten van het programma naar een andere machine lastig, zo niet ondoenlijk is.

In dit hoofdstuk hopen we u antwoord te kunnen geven op de volgende vragen:

1. Hoe vind ik 'mooie' driedimensionale functies?

2. Hoe ziet een algemeen programma voor het tekenen van een dergelijke functie eruit?

3. Hoe onderdruk je het tekenen van de onzichtbare lijnen (hidden lines)?

We beginnen met de definitie van een driedimensionale functie.

Elke functie van de vorm $z = f(x,y)$ heet een driedimensionale functie en vormt een, meestal, gekromd vlak in een driedimensionaal (ruimtelijk) coördinatenstelsel. Speciaal voor niet-wiskundigen volgt nu een voorbeeld van een functie $z = f(x,y)$ en het gekromde vlak dat als 'grafiek' bij de functie hoort.

Stel dat we voor $z = f(x,y)$ de volgende vergelijking nemen:

$$z = e^{-(x^2+y^2)}$$

(e is het grondtal van de natuurlijke logaritme; e = 2,718284183...)

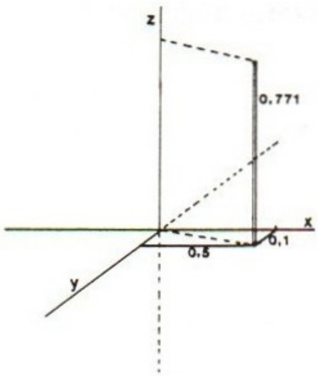
Voor elk punt (x,y) in het (platte) X-Y-vlak kunnen we nu de daarbij horende waarde van z uitrekenen. Als x = 0,5 en y = 0,1 dan is

$z = e^{-(0,25+0,01)} \Rightarrow$

$z = e^{-0,26} \Rightarrow$

z = 0,771, want 2,71828... tot de macht -0,26 is 0,771.

Zet nu (in gedachten) in het voetpunt P(0,5;0,1) in het X-Y-vlak een staafje met een lengte van 0,771 rechtop. Doe ditzelfde voor nog een groot aantal punten (x,y).




Leg nu over al deze staafjes een elastische folie. Dit (golvende) stuk folie vormt een goed model van het vlak met vergelijking $z = e^{-(x^2+y^2)}$.
Hoe dit eruit zou zien kunt u zien op p.73.

## Antwoord op de eerste vraag

Zoek met behulp van programma 7 uit hoofdstuk 2 een willekeurige contine functie die symmetrisch ten opzichte van de y-as is. De grafiek van de functie moet 'bergen en dalen' vertonen (de functie moet maxima en minima hebben). Erg geschikt zijn trigonometrische functies (sin, cos) en combinaties van deze functies. Ook geschikt zijn functies waarin alleen termen voorkomen met 'even-machten' van x en exponentiële functies.

Enkele voorbeelden:

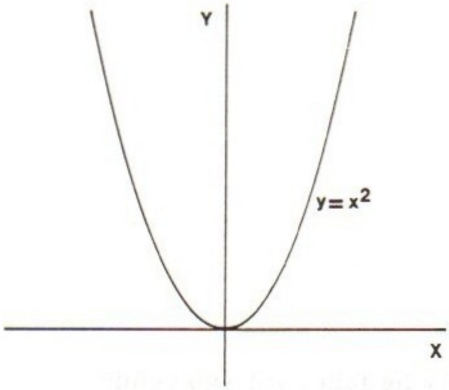
$y = e^{-x^2}$ ; $y = \sin(x)$

$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7}$$

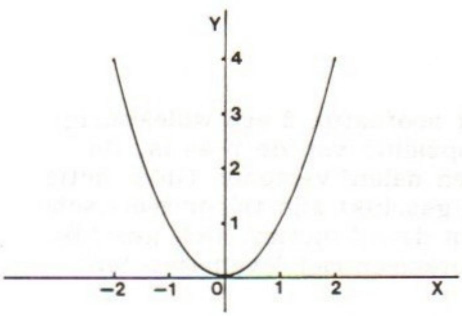
Bekijk nu de grafiek van zo'n functie in het interval $-a \leq x \leq a$. Dit 'stuk grafiek gaan we draaien rond de y-as. Zo ontstaat een, ten opzichte van de y-as, draaisymmetrisch driedimensionaal vlak. Als we nu de y-as als z-as kiezen, dan wordt de vergelijking van een dergelijk draaisymmetrisch ruimtelijk vlak (rond de z-as) van de vorm

$$z = f(r) \qquad \text{met} \quad r = \sqrt{x^2 + y^2}$$

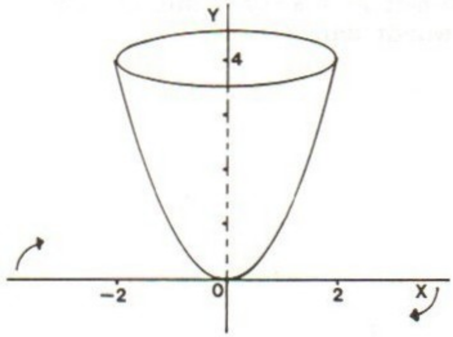
We zullen dit aan een eenvoudig voorbeeld proberen te verklaren. We kiezen als voorbeeld de eenvoudige paraboolvergelijking $y=x^2$.



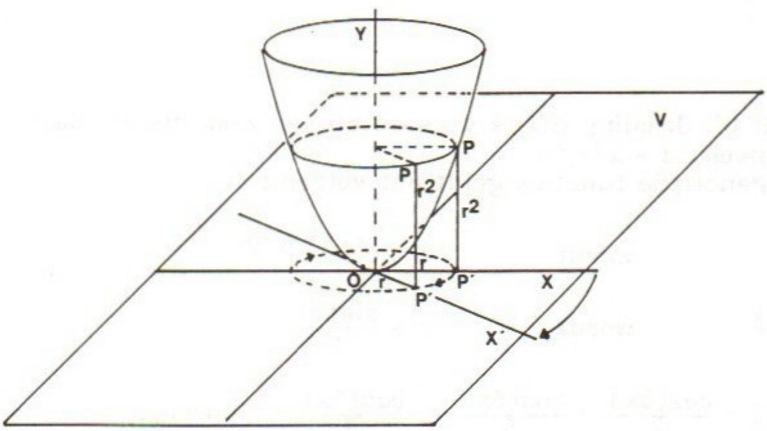

Stel we bekijken de grafiek voor $-2 \leq x \leq 2$:




Als we nu de x-as rond de y-as laten draaien (de x-as komt als het ware loodrecht het papier uit), dan ontstaat een draaisymmetrisch vlak rond de y-as. Dit is een kegel:




Elk punt P op de kegelmantel heeft de eigenschap $y=r^2$, waarbij r de afstand is van het geprojecteerde punt P' tot het punt O. We kunnen het vlak V dat door de ronddraaiende x-as wordt gevormd als volgt tekenen:

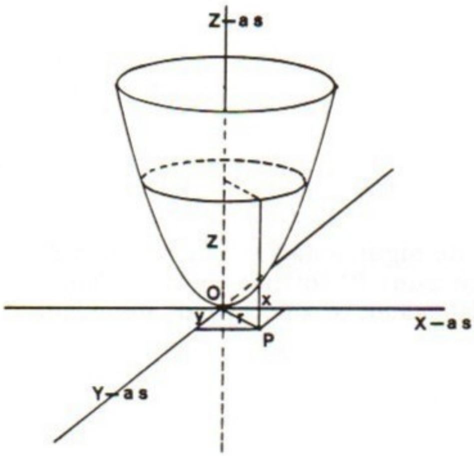


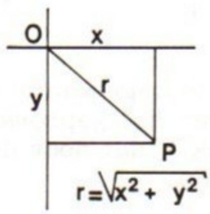

Welnu, als we nu de z-as als y-as nemen en we nemen de y-as in vlak V loodrecht op de x-as, dan zien we dat de afstand r geschreven kan worden als

$r = \sqrt{x^2+y^2}$ (stelling van Pythagoras, zie rechts in de volgende tekening)

en nu wordt $y=r^2$ geschreven als $z=r^2$ met $r^2 = x^2+y^2$, dus de vergelijking van de paraboloïde (kegel) wordt dan

$$z = x^2+y^2$$




Dus $y=x^2$ wordt bij draaiing om de y-as en bij een z-as die de plaats van de y-as inneemt $z = x^2+y^2$.
Voor de bovengenoemde functies geldt het volgende:

$$y = e^{-x^2} \quad \text{wordt} \quad z = e^{-(x^2+y^2)}$$

$$y = \frac{\sin(x)}{x} \quad \text{wordt} \quad z = \frac{\sin(r)}{r}$$

$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7} \quad \text{wordt}$$

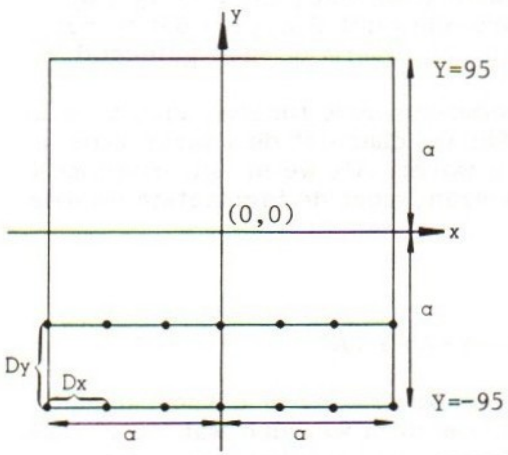
$$y = \cos(r) - \frac{\cos(3r)}{3} + \frac{\cos(5r)}{5} - \frac{\cos(7r)}{7}$$

met $r = \sqrt{x^2+y^2}$

Voor deze klasse van draaisymmetrische figuren ontwikkelen we een algemeen tekenprogramma.

## Antwoord op de tweede vraag

Bekijk de onderstaande tekening. U kijkt recht bovenop het ruimtelijke vlak van een driedimensionale functie. We willen dat het op het grondvlak geprojecteerde vlak een vierkant is met zijden van 2a. Straks zullen we zien dat, als we een mooie tekening van zo'n ruimtelijk vlak op het beeldscherm willen maken, a=95 een heel goede keus is.




Als 'doorgewinterde' programmeur ziet u natuurlijk direct dat het programma voor het tekenen van het vlak een geneste FOR-NEXT (dx,dy) zal bevatten! Als we een vlak als grafiek van z = f(x,y) willen tekenen, moeten we namelijk voor een aantal combinaties van x en y de functiewaarde z = f(x,y) berekenen. We beginnen met y=-95 en laten x met stapjes dx (bijvoorbeeld dx=3) van -95 naar +95 lopen. Voor elke combinatie x,y met y=-95 berekenen we de functiewaarde z = f(x,y). Zo ontstaan de punten (x,y,z) van het ruimtelijke vlak waarvan de geprojecteerde punten (x,y) in de bovenstaande tekening op de onderste zijde van het vierkant als bolletjes getekend zijn. Nu verhogen we y met dy en laten x weer van -95 tot +95 lopen. Dit geeft de tweede reep van het vlak. Op deze wijze ontstaat het hele vlak, waarvan de punten (x,y,z) natuurlijk nog getransformeerd moeten worden naar beeldschermcoördinaten (x',y').

Een eerste globale opzet voor het tekenen van een 'vierkant'-stuk vlak is:

```
FOR Y=-95 TO 95 STEP DY
   FOR X=-95 TO 95 STEP DX
      GOSUB 1000
      :
   NEXT X
NEXT y
```

In de subroutine 1000 wordt voor een punt (x,y) de waarde van z = f(x,y) berekend. Met de bekende transformaties wordt vervolgens het punt P(x,y,z) overgebracht naar een punt P'(x',y') op het beeldscherm. Nu kan het berekende punt P(x,y,z) dat in het vlak ligt, als het punt P'(x',y') op het scherm worden getekend.

Jammer genoeg lenen niet alle driedimensionale functies zich voor de intervallen $-95 \leq x \leq 95$ en $-95 \leq y \leq 95$; vandaar dat de waarde voor a met een INPUT-opdracht ingelezen wordt. Als we in het programma toch $-95 \leq XX \leq 95$ en $-95 \leq YY \leq 95$ kiezen, moet de ingetoetste waarde A als volgt gebruikt worden:

$$X = XX.\frac{A}{95} \quad \text{en} \quad Y = YY.\frac{A}{95}$$

dat wil zeggen er geldt: $-A \leq X \leq A$ en $-A \leq Y \leq A$

De z-waarden van met name de trigonometrische driedimensionale functies zal tussen -1 en 1 liggen. Om deze waarden wat 'op te blazen' kan een vermenigvuldigingsfactor (K1) worden ingetoetst. Goede waarden voor K1 liggen tussen 30 en 80.

We kunnen ons programma nu als volgt opschrijven:

```
opzet voor een programma voor
    het tekenen van de functie
       Z=f(X,Y)

    variabelen waarde geven en
    hoge resolutie instellen
 FOR YY=-95 TO 95 STEP DY
  Y=YY*AF
  FOR XX=-95 TO 95 STEP DX
   X=XX*AF
   GOSUB 1000 'functiewaarde Z
   XG=INT(U+XX+C*YY+H)
   YG=INT(V-S*YY-Z +H)

      teken punt P(XG,YG)

  NEXT XX
 NEXT YY

    programma afsluiten

  REMbeschrijving van de functie
         Z=f(X,Y)
```

Zoals beloofd laten we nu zien waarom de intervallen $-95 \le XX \le 95$ en $-95 \le YY \le 95$ zo geschikt zijn om driedimensionale functies te tekenen met een hoog oplossend vermogen. We willen graag de fraaie projectie met $\alpha=45°$ en k=0,5 gebruiken. Wat worden dan de beeldschermcoördinaten XG en YG voor de hoekpunten 'linksonder' en 'rechtsboven' van het vierkant op p.69? Deze punten bepalen immers of de hele grafiek op ons beeldscherm van 256 bij 192 puntjes getekend kan worden. Voor K1 nemen we 50, dat ligt zo'n beetje tussen 30 en 80! Nemen we voor de z-waarde ook 50, dan gaat het punt (x,y,z) met coördinaten (-95,-95,50) over in het beeldschermpunt (XG,YG) met

XG = INT(128-95-0,5.95.cos(45)+0,5) ⇒ = XG = -1
YG = INT(96+0,5.95.sin(45)-50+0,5) ⇒ = YG = 80

De YG-waarde ligt tussen 0 en 192, maar de XG-waarde ligt net buiten het scherm. Voor dit hoekpunt is dit echter geen bezwaar, het ligt immers in de uiterste hoek van onze tekening! De 'breedte' (256) van het scherm wordt heel goed benut, kijk maar eens naar de coördinaten (XG,YG) voor het punt (95,95,50) rechtsboven:

XG = INT(128+95+0,5.95.cos(45)+0,5) ⇒ 257
YG = INT(96-0,5.95.sin(45)-50+0,5) ⇒ 12

We benutten dus ruimschoots de hele schermbreedte ($-1 \le XG \le 257$) voor het tekenen van de driedimensionale figuur. In programma 28 (regel 300) en in programma 29 (regels 330 en 340) is rekening gehouden met eventuele negatieve XG-waarden en XG-waarden groter dan 255. Als dit voorkomt, worden deze waarden op respectievelijk 0 en 255 gezet, zodat het programma niet door een foutmelding afbreekt.

Op de onder- en bovenrand (YY=-95 en YY=+95) gelden vaak heel kleine z-waarden, zodat bij $\alpha=45°$, k=0,5 en z=0 de grafiek op het scherm zal liggen tussen

↓ z-waarde
YG = INT(96+0,5.95.sin(45)-0+0,5 ⇒ 130
YG = INT(96-0,5.95.sin(45)-0+0,5 ⇒ 62

Ons tekenvlak is dus ongeveer $-1 \le XG \le 257$ èn $62 \le YG \le 130$

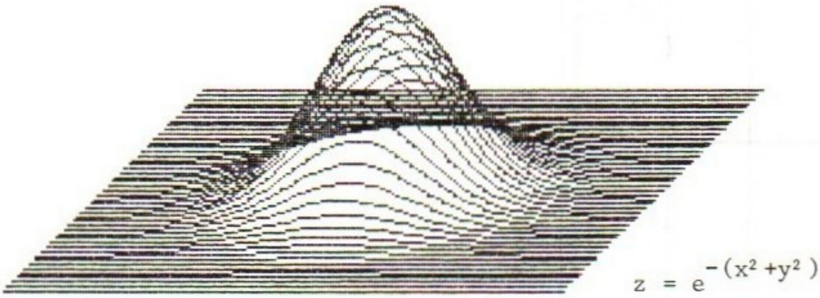
In het bovenstaande programmavoorstel wordt gebruik gemaakt van 'puntgraphics'. Hierbij worden de punten puntje voor puntje getekend. Om een enigszins acceptabele tekening te krijgen moeten erg veel 'puntjes' berekend worden (DX=1,DY=1). Dit duurt in BASIC (geneste lussen en vele trigonometrische berekeningen) erg lang.

Voor de onderstaande tekeningen betekent dit al snel een tekentijd van meer dan 30 minuten, soms wel een uur. Dit is weinig bevredigend! Veel sneller gaat het als we de 'vectorgraphics'-methode gebruiken. Deze techniek hebben we in alle voorgaande programma's gebruikt; we verbinden steeds twee naast elkaar liggende punten door een recht lijntje. Hierbij hoeven we lang niet zoveel punten te berekenen als bij de 'puntgraphics'-techniek. Bovendien ziet de tekening er beter uit.

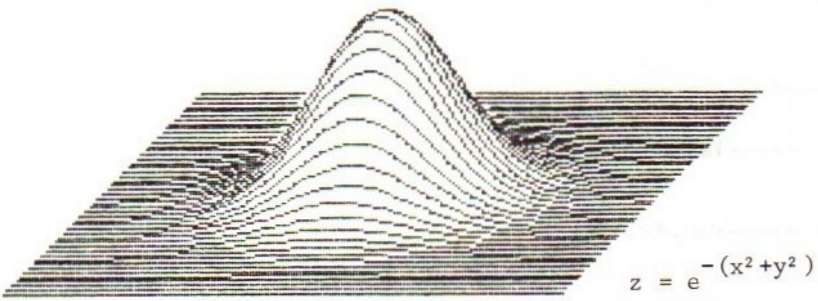
Als antwoord op de tweede vraag komen we dan met het volgende algemene programma voor het tekenen van draaisymmetrische ruimtelijke vlakken.

```
100 REM programma 28 grafiek van de
                        functie Z=f(X,Y)
110 CLS
120 INPUT "alfa in grad.(45-135) ";W
130 INPUT "verkl.factor(.5-.75) ";K
140 INPUT "rechtergrens x (>0) ";A
150 INPUT "vergr.factor functie Z";K1
160 PI=4*ATN(1)
170 U=128:V=96:H=.5:RD=PI/180
180 C=K*COS(W*RD) : S=K*SIN(W*RD)
190 DX=3 : DY=5 : AF=A/95
200 CLS
210 COLOR 1,15,4 : SCREEN 2
220 FOR YY=-95 TO 95 STEP DY
230  Y=YY*AF
240  FOR XX=-95 TO 95 STEP DX
250   X=XX*AF
260   GOSUB 1000 'functiewaarde Z
270   XG=INT(U+XX+C*YY+H)
280   YG=INT(V-S*YY-Z +H)
290   IF XG<0 THEN XG=0
300   IF XG>255 THEN XG=255
310   IF YG<0 OR YG>191
        THEN PRINT "foute K1": STOP
320   IF XX=-95
        THEN X1=XG:Y1=YG
        ELSE X2=XG:Y2=YG:
             LINE(X1,Y1)-(X2,Y2):
             X1=X2:Y1=Y2
330  NEXT XX
340 NEXT YY
350 A$=INPUT$(1)
360 COLOR 15,4,4 : END
370 '
1000 REM subroutine functiewaarde Z
1010  Z=K1*EXP(-X*X-Y*Y)
1020 RETURN
```

Met dit programma maakt u de onderstaande tekening. Kies op uw MSX-computer de volgende waarden: W=45°, K=0,5, A=3 en K1=70.




U zult over de bovenstaande tekening niet tevreden zijn. De voorgrond is goed, maar in het achterste stuk zijn ook alle lijnen die voor ons onzichtbaar zijn getrokken. In onderstaande tekening zien we het resultaat met dezelfde waarden voor W, K, A en K1, maar waarin de niet-zichtbare lijnen ook niet getekend zijn. Hoe krijgen we dit voor elkaar?

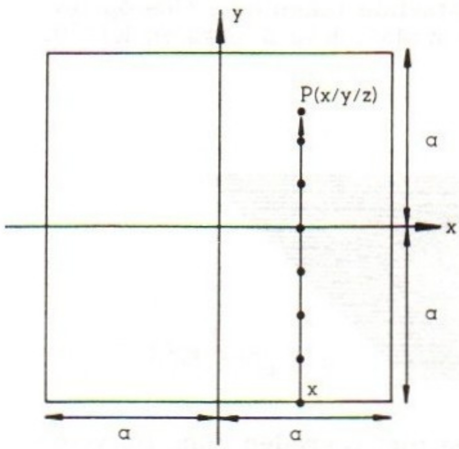



## Antwoord op de derde vraag

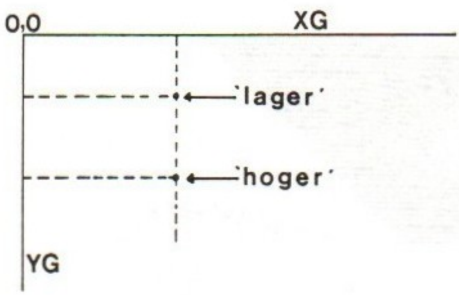
Bekijk de onderstaande tekening. Een punt P(x,y,z) van het te tekenen vlak is alleen dan zichtbaar wanneer:

a. het punt 'hoger' ligt dan alle voorgaande punten met dezelfde x-coördinaat of
b. het punt 'lager' ligt dan alle voorgaande punten met dezelfde x-coördinaat.

Vanzelfsprekend geldt dit ook voor de op het beeldscherm geprojecteerde punten P(XG,YG).




Bedenk bij de navolgende uiteenzetting dat de coördinaten (XG,YG) van een beeldschermpunt 'gemeten' worden ten opzichte van de linkerbovenhoek van het beeldscherm, de oorsprong van ons HRG-coördinatenstelsel.




Een punt, in beeldschermcoördinaten uitgedrukt, ligt dus 'hoger' dan eenander punt (bij dezelfde XG-waarde) als de YG-waarde groter is en ligt 'lager' als de YG-waarde kleiner is.

We gaan als volgt te werk. We gebruiken twee arrays (H1(XG) en H2(XG)) waarin we voor alle XG-coördinaten in ons vlak de respectievelijk kleinste en grootste YG-waarde bewaren. Vinden we voor een punt met een bepaalde XG-waarde een bijbehorende YG-waarde die òf kleiner is dan H1(XG) òf groter is dan H2(XG), dan is het punt zichtbaar en wordt H1(XG) of H2(XG) gelijk gemaakt aan deze

nieuwe kleinste of grootste YG-waarde. Geldt voor een bepaalde XG dat H1(XG)<YG<H2(XG), dan is het punt (XG,YG) niet zichtbaar. H1(XG) en H2(XG) zijn te zien als twee horizonnen.

Als voor een bepaald punt $P_1(x,y,z)$ uit het vlak het beeldpunt P1(XG,YG) berekend is, gaan we kijken of YG kleiner dan of gelijk aan H1(XG) is (YG<=H1(XG)). Is dit zo dan is het punt P1 op het beeldscherm zichtbaar. Nu zetten we de vlag F1 op 1 en we maken H1(XG) gelijk aan YG (H1(XG)=YG). Is dit niet het geval (YG> H1(XG)), dan is P1 onzichtbaar en blijft de vlag F1 op 0 staan.

Nu gaan we naar het volgende punt $P_2(x,y,z)$ uit het vlak (we verhogen x met dx). Opnieuw berekenen we het beeldpunt P2(XG,YG). Weer wordt gekeken of YG<=H1(XG) en zonodig wordt de vlag F2 op 1 gezet. De punten P1 en P2 worden alleen dan door een rechte lijn (lijntje) verbonden als beide vlaggen F1 en F2 de waarde 1 hebben, dus als F1xF2=1.

Ditzelfde doen we voor de tweede 'horizon' H2(XG). We kijken of YG>=H2(XG), ....., enzovoorts.

Omdat we bij een vaste y-waarde de x-waarde steeds met dx ophogen (bijvoorbeeld dx=3) en hiermee steeds twee buurpunten P1(XG,YG) en P2(XG,YG) berekenen slaan we als het ware een aantal punten over waarvoor geen H1(XG)- en H2(XG)-waarden berekend worden. Door lineaire interpolatie tussen de horizonwaarden (H1(XG) en H2(XG)) van twee buurpunten zouden we de horizonwaarden voor de hiertussenliggende punten kunnen berekenen. We hebben deze waarden wel nodig, omdat het kan voorkomen dat, als we y met dy verhogen, we XG-waarden vinden waarvoor nog geen H1(XG)- en H2(XG)-waarden berekend zijn.

Een programma dat de hierboven geschetste werkwijze zou volgen zou, in BASIC geschreven, veel te langzaam zijn. Om de tekensnelheid acceptabel te houden, dat wil zeggen niet langer dan 10 minuten tekenen voor één figuur, passen we de volgende vereenvoudigingen toe:

1. We berekenen alleen de onderste (voor de kijker de bovenste) horizon H1(XG).

2. We passen geen lineaire interpolatie toe bij het berekenen van de array-waarden in H1(XG). Alle beeldpunten in een interval ter lengte dx krijgen dezelfde H1(XG)-waarde.

Als we de stapgrootte dx maar klein genoeg kiezen (een goede waarde is dx=3), krijgen we ondanks deze twee simplificaties toch accep-

tabele tekeningen. (Bekijk de volgende tekeningen en oordeelt u zelf.) Het niet-tekenen van 'voor de kijker' onzichtbare punten komt tot stand door het volgende stukje programma:

```
F1=0:L=INT(XG/DX)
IF YG<=H(L) THEN F1=1:H(L)=YG
```

Laten we dit met een voorbeeld illustreren. Stel we berekenen van een punt P(x,y,z) in het ruimtelijke vlak de projectie op het beeldscherm. Dit gebeurt door eerst met x en y de z-waarde te berekenen. (In programma 29 krijgt Y in regel 270, X in regel 290 en Z in regel 1010 een waarde.) Als de x-, y- en z-waarde van een punt in het vlak berekend zijn, wordt dit punt op het beeldscherm geprojecteerd door het berekenen van de twee coördinaten XG en YG. (In programma 29 gebeurt dit in regel 310-320.) Stel nu dat als beeldpunt het punt (XG=40 en YG=126) berekend is. Het programma zal de vlag F1 op nul zetten (regel 2010) en de waarde L=INT(XG/DX) berekenen. Voor XG=40 en DX=3 wordt deze waarde INT(40/3), dus L=13. Dit betekent in feite dat de drie punten met XG=39, XG=40 en XG=41 dezelfde horizonwaarde L=13 opleveren. Mocht nu eerder in H(13) als horizonwaarde de waarde 132 opgeslagen zijn, dan is YG<=H(L) (regel 2020) waar, want 126<=132 is waar, en is het punt (40,126) zichtbaar. Op dit moment wordt de vlag F1 gezet en wordt YG=126 de nieuwe horizonwaarde bij L=13 (regel 2020).

De regels

```
200 DIM H(255)
210 FOR L=0 TO 255
220     H(L)=1000
230 NEXT L
```

zorgen ervoor dat, voordat het tekenen begint, het hele scherm 'zichtbaar' is. Dit is zo als de onderste horizon (voor de kijker de bovenste horizon) de onderrand van het beeldscherm is. De waarde 1000 geeft in feite een horizon die 'ver onder' het beeldscherm ligt. Op p.77 staat het programma dat alle voor ons onzichtbare punten ook niet tekent.

Met W=45°, K=0,5, A=3 en K1=80 krijgt u de grafiek van $z=e^{-(x^2+y^2)}$ zoals die een paar pagina's eerder is getekend; de niet-zichtbare punten die achter de hoed liggen zijn inderdaad niet te zien!

```
100 REM programma 29 grafiek van de
      functie Z=f(X,Y), hidden lines
110 CLS
120 INPUT "alfa in grad.(45-135) ";W
130 INPUT "verkl.factor(.5-.75)  ";K
140 INPUT "rechtergrens x (>0)   ";A
150 INPUT "vergr.factor functie Z";K1
160 PI=4*ATN(1)
170 U=128:V=96:H=.5:RD=PI/180
180 C=K*COS(W*RD) : S=K*SIN(W*RD)
190 DX=3 : DY=5 : AF=A/95
200 DIM H(255)
210 FOR L=0 TO 255
220    H(L)=1000
230 NEXT L
240 CLS
250 COLOR 1,15,4 : SCREEN 2
260 FOR YY=-95 TO 95 STEP DY
270  Y=YY*AF
280  FOR XX=-95 TO 95 STEP DX
290   X=XX*AF
300   GOSUB 1000 'functiewaarde Z
310   XG=INT(U+XX+C*YY+H)
320   YG=INT(V-S*YY-Z +H)
330   IF XG<0 THEN XG=0
340   IF XG>255 THEN XG=255
350   IF YG<0 OR YG>191
        THEN PRINT "foute K1": STOP
360   IF XX=-95
             THEN GOSUB 2000
             ELSE GOSUB 3000
370  NEXT XX
380 NEXT YY
390 A$=INPUT$(1)
400 COLOR 15,4,4 : END
410 '
1000 REM subroutine functiewaarde Z
1010  Z=K1*EXP(-X*X-Y*Y)
1020 RETURN
2000 REM subroutine eerste xx-waarde
2010  F1=0 : L=INT(XG/DX)
2020  IF YG<=H(L) THEN F1=1:H(L)=YG
2030  X1=XG:Y1=YG
2040 RETURN
3000 REM subroutine volgende xx'en
3010  F2=0 : L=INT(XG/DX)
3020  IF YG<=H(L) THEN F2=1:H(L)=YG
3030  X2=XG:Y2=YG
3040  IF F1*F2=1
        THEN LINE (X1,Y1)-(X2,Y2)
3050  X1=X2:Y1=Y2
3060 RETURN
```

Hoe kleiner we DX en DY maken, des te gedetailleerder zal de figuur worden. Maar ook 'des te langer zal het tekenen duren'. De combinatie DX=3 en DY=5 is een goed compromis tussen de tekensnelheid en de mate van gedetailleerdheid.

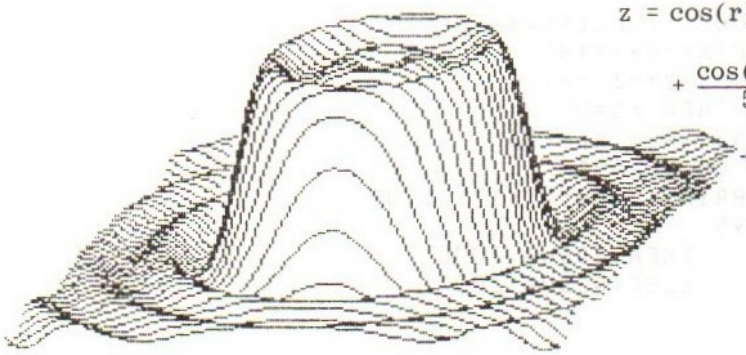
Dit programma is een algemeen programma voor het tekenen van draaisymmetrische ruimtelijke vlakken. Als u andere figuren wilt maken, verander dan alleen de functie in de subroutine 1000 en kies mogelijk andere waarden voor de variabelen.

Nu volgt een aantal voorbeelden.

```
1000 REM subroutine functiewaarde Z
1010   R=SQR(X*X+Y*Y)*RD
1020   Z=K1*(COS(R)-COS(3*R)/3+
             COS(5*R)/5-COS(7*R)/7)
1030 RETURN
```

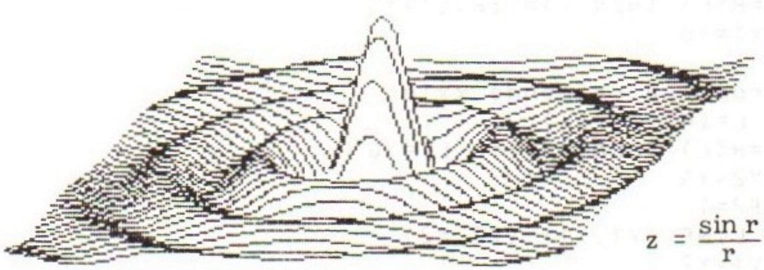
U kiest: W=45, K=0,5, A=180°, K1=45.

$$z = \cos(r) - \frac{\cos(3r)}{3} + \frac{\cos(5r)}{5} - \frac{\cos(7r)}{7}$$



```
1000 REM subroutine functiewaarde Z
1010   R=SQR(X*X+Y*Y)*RD
1020   IF R=0 THEN Z=K1
             ELSE Z=K1*SIN(R)/R
1030 RETURN
```

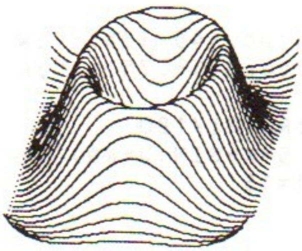
U kiest: W=45, K=0,5, A=1080, K1=60.



$$z = \frac{\sin r}{r}$$

```
1000 REM subroutine functiewaarde Z
1010 R=SQR(X*X+Y*Y)
1020  Z=K1*EXP(-COS(R/16))
1030 RETURN
```

U kiest: W=45, K=0,5, A=80, K1=30.

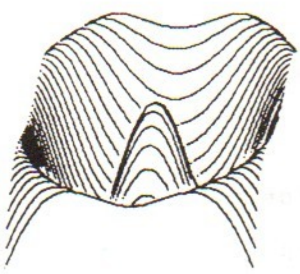


$z = \exp(-\cos(\frac{r}{16}))$

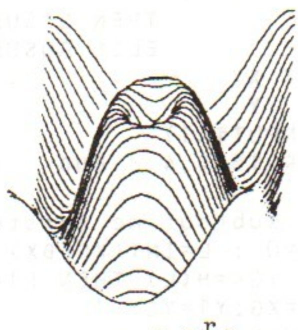
Voor de volgende twee tekeningen geldt:

```
1000 REM subroutine functiewaarde Z
1010  R=SQR(X*X+Y*Y)
1020 Z=K1*COS(R/16) EN Z=K1*SIN(R/16)
1030 RETURN
```

U kiest: W=45, K=0,5, A=90, K1=50.



$z = \cos(\frac{r}{16})$



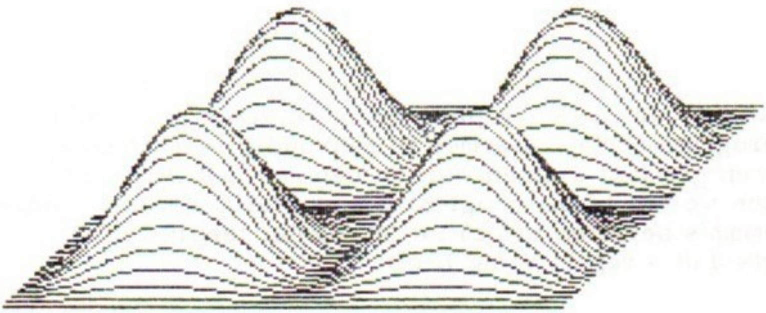
$z = \sin(\frac{r}{16})$

Met dit programma hebben we talrijke andere grafieken gemaakt. Elke keer dat u een 'leuke functie' tegenkomt, kunt u meteen de daarbij horende draaisymmetrische figuur maken.

Tot slot volgt nog een programma voor het tekenen van een fraaie vier-toppige grafiek. U bent hem misschien wel eens in een computertijdschrift tegengekomen. Om de snelheid op te voeren is de subroutine 1000 in het hoofdprogramma opgenomen. U kunt dit natuurlijk ook in alle andere programma's doen.

```
100 REM programma 30 mooie functie
110 CLS
120 INPUT "alfa in grad.(45-135) ";W
130 INPUT "verkl.factor(.5-.75)  ";K
140 PI=4*ATN(1)
150 U=128:V=96:H=.5:RD=PI/180
160 C=K*COS(W*RD) : S=K*SIN(W*RD)
170 DX=3 : DY=5 : K1=15
180 DIM H(255)
190 FOR L=0 TO 255
200    H(L)=1000
210 NEXT L
220 CLS
230 COLOR 1,15,4 : SCREEN 2
240 FOR YY=-95 TO 95 STEP DY
250  M1=COS(YY*2*PI/95-PI)+1
260  FOR XX=-95 TO 95 STEP DX
270   M2=COS(XX*2*PI/95-PI)+1
280   Z=K1*M1*M2
290   XG=INT(U+XX+C*YY+H)
300   YG=INT(V-S*YY-Z +H)
310   IF XG<0 THEN XG=0
320   IF XG>255 THEN XG=255
330   IF XX=-95
                THEN GOSUB 1000
                ELSE GOSUB 2000
340  NEXT XX
350 NEXT YY
360 A$=INPUT$(1)
370 COLOR 15,4,4 : END
380 '
1000 REM subroutine eerste xx-waarde
1010  F1=0 : L=INT(XG/DX)
1020  IF YG<=H(L) THEN F1=1:H(L)=YG
1030  X1=XG:Y1=YG
1040 RETURN
2000 REM subroutine volgende xx'en
2010  F2=0 : L=INT(XG/DX)
2020  IF YG<=H(L) THEN F2=1:H(L)=YG
2030  X2=XG:Y2=YG
2040  IF F1*F2=1
        THEN LINE (X1,Y1)-(X2,Y2)
2050  X1=X2:Y1=Y2:F1=F2
2060 RETURN
```

Voor de onderstaande tekening kiest u W=45°, K=0,5 en K1=15 of 20.



$$z = (\cos(\frac{2\pi x}{95} - \pi)+1(\cos(\frac{2\pi y}{95} - \pi)+1)$$

# 6 Turtle-graphics en LOGO-simulatie

LOGO is vooral door de turtle-graphics beroemd geworden. Er is geen andere programmeertaal waarmee zo eenvoudig zeer moeilijke grafische figuren gemaakt kunnen worden dan met LOGO. In dit hoofdstuk zullen we vijf LOGO-programma's in BASIC vertalen. Deze BASIC-programma's bevatten wel 25 tot 30 regels, terwijl LOGO hiervoor slechts 3 of 4 regels nodig heeft.

"Eerst LOGO, daarna andere programmeertalen"

is het motto van vele informatici en pedagogen die zich met de invoering van de informatica op scholen bezighouden.

Wanneer men deze stelling wil begrijpen, moet men niet alleen de taal LOGO en haar mogelijkheden, maar ook de omgeving waarin LOGO ontwikkeld werd goed kennen. Een korte historische terugblik lijkt daarom op zijn plaats.

Het schrijven van computerprogramma's is een intellectuele prestatie en een creatief proces. Aan de wijze waarop wij programma's ontwikkelen, herkennen wij direct onze manier van denken. Aan het Massachusetts Institute of Technology (MIT) heeft men al in het begin van de zestiger jaren een speciale programmeertaal ontwikkeld, met behulp waarvan men kunstmatige intelligentie ging bestuderen. Men noemde deze taal LISP, een afkorting van LISt-Processing.

In LISP werden programma's geschreven die een met kunstmatige zintuigen uitgeruste elektromechanische muis in staat stelden een uitweg uit ieder willekeurig doolhof te vinden. In LISP worden programma's ontwikkeld die de computer tot een medespeler met leervermogen maakt. Hoe meer spelletjes de computer tegen een menselijke tegenspeler speelt, des te beter wordt hij. Goede zetten van de tegenstander onthoudt hij en legt hij vast in zijn geheugen, terwijl hij de eigen slechte zetten voor toekomstige spelletjes uit zijn

geheugen wegveegt. Een 'afvalprodukt' van deze studies aan het MIT zijn de moderne schaakprogramma's.

LOGO is een hoog ontwikkeld dialect van de LISP-taal. Seymour Papert, leerling van de bekende onderzoeker Jean Piaget en medewerker aan het MIT, heeft in twaalf jaar tijd LOGO ontwikkeld. Jean Piaget heeft in zijn bekende boek *Hoe kinderen leren* de kinderlijke denkstructuren laten zien. Hieruit blijkt dat tekenen en knutselen tot de eerste creatieve handelingen van kinderen behoren. Wij zullen zien hoe LOGO deze bezigheden ondersteunt.

Amerikanen gaan door voor een volk dat graag experimenteert. Het was hun echter duidelijk, dat jonge kinderen, wij denken dan aan zes- tot dertienjarigen, niet in staat zijn een programmeertaal zoals BASIC, laat staan Pascal, te leren en algorithmen voor numerieke, niet-numerieke of grafische problemen te schrijven. Daarom heeft men een 'kinderlijke' taal gemaakt (waarachter echter een imponerende software schuilgaat), waarmee het kind met gemak tekeningen kan maken.

Wordt het LOGO-systeem ingeschakeld, tegenwoordig meestal een 64K-microcomputer met bijbehorende software, dan verschijnt in het midden van het beeldscherm een kleine driekhoek, waarvan de top naar boven wijst. De kinderen noemen die driehoek 'turtle', het Engelse woord voor schildpad. Met behulp van eenvoudige bevelen kan het kind de schildpad willekeurig over het beeldscherm laten rondlopen. En al naar gelang het wenselijk is laat de turtle wel of geen spoor na.

De volgende LOGO-opdrachten zijn beschikbaar:

| | | |
|---|---|---|
| FORWARD | 100 | = 100 passen vooruit |
| BACK | 50 | = 50 passen achteruit |
| RIGHT | 90 | = draaiing van 90° met de klok mee |
| LEFT | 45 | = draaiing van 45° tegen de klok in |
| PENUP | | = haal pen van papier |
| PENDOWN | | = zet pen op papier |
| HIDETURTLE | | = haal Turtle van het beeldscherm |

Normaal gesproken geldt de toestand "PENDOWN". Als we bijvoorbeeld de hoofdletter F op het scherm willen tekenen, kan dat in LOGO als volgt:

```
FORWARD 100 RIGHT 90 FORWARD 50
RIGHT 90 PENUP FORWARD 50 PENDOWN
RIGHT 90 FORWARD 50
HIDETURTLE
```

Voor 100, 50, 90 en 45 kunnen ook andere waarden gekozen worden.

Omdat LOGO een vertolkend systeem is, wordt elke opdracht (na het geven van RETURN) direct uitgevoerd. Zo op het oog lijkt het slechts leuk speelgoed! Laten we nu eens een LOGO-programma bekijken; liever spreken we van een procedure:

```
TO VIERKANT
REPEAT 4 (FORWARD 75 RIGHT 90)
END
```

Het LOGO-systeem weet dat tussen TO en END een procedure (een stuk programma) gedefinieerd wordt. De procedure tussen TO en END (in ons voorbeeld VIERKANT genoemd) maken we zelf. Zouden we bovenstaande procedure intoetsen, dan zal op het beeldscherm een vierkant met zijden ter lengte 75 getekend worden. De programmeurs onder u kennen vast de opdracht REPEAT waarmee een herhalingsstructuur geprogrammeerd kan worden.

Als we de procedure VIERKANT in het LOGO-systeem ingevoerd hebben, kunnen we hier ook gebruik van maken. LOGO onthoudt alle procedures die wij zelf invoeren. We kunnen dan ook later nieuwe procedures ontwikkelen waarin we gebruik maken van eerder ingevoerde procedures (zoals VIERKANT). Nu volgt een procedure met een parameter:

```
TO VIERKANT:ZIJDE
REPEAT 4 (FORWARD:ZIJDE RIGHT90)
END
```

Toetsen we nu VIERKANT 150 <RETURN> in, dan zal LOGO een vierkant met zijden van 150 tekenen. Niets verhindert u een procedure in te bedden in een volgende procedure en die weer in een volgende en die weer ....., enzovoorts. Alleen de beschikbare geheugenruimte is hierbij een remmende factor. Bekijk het volgende LOGO-programma.

LOGO-programma nr.1

```
TO VIERKANTPATROON
REPEAT 8 (FORWARD 20 LEFT 45 VIERKANT 75)
END
```

Dit eenvoudige drieregelige LOGO-programma tekent een patroon van acht vierkanten (zie p.87). We zullen dit programma straks in BASIC vertalen. Het zal blijken dat hiervoor enige wiskundige en programmeertechnische kennis nodig is. Het LOGO-programma kan door een leerling van de lagere school gemaakt worden, terwijl het

BASIC-programma dat deze acht vierkanten tekent slechts door scholieren van de hoogste klassen van het voortgezet onderwijs kan worden gemaakt. Het wordt nog moeilijker als we intikken:

```
TO STROOK:AANTAL
REPEAT:AANTAL (FORWARD 100  VIERKANTPATROON)
END
```

Tikken we vervolgens in STROOK 5 <RETURN> dan maken we heel eenvoudig een fraai ornament. Probeer dit maar eens in BASIC of Pascal. LOGO wordt pas echt interessant als we van de mogelijkheid gebruik maken dat een procedure zichzelf aanroept (recursiviteit). De turtle-graphics berusten op het principe dat we een procedure parameters geven en dat deze procedure zichzelf steeds met andere parameterwaarden aanroept. Bekijk de volgende procedure:

```
TO VEELHOEK:ZIJDE:HOEK
FORWARD:ZIJDE  LEFT:HOEK
VEELHOEK:ZIJDE:HOEK   <--   hier roept de procedure
END                         zichzelf aan
```

Als we nu VEELHOEK 200 144 intikken, zal het LOGO-systeem een regelmatige vijfpuntige ster tekenen. We zullen echter op de STOP-toets moeten drukken om het tekenen te stoppen. Brengen we in deze procedure een stopmechanisme aan en laten we de procedure steeds de waarde van ZIJDE veranderen, dan krijgen we de welhaast bekendste LOGO-procedure:

LOGO-programma nr.2

```
TO TURTLE:ZIJDE:GROEI:HOEK
FORWARD:ZIJDE  LEFT:HOEK
MAKE:ZIJDE:ZIJDE+:GROEI
IF:ZIJDE>200 THEN STOP
TURTLE:ZIJDE:GROEI:HOEK
END
```

Als er een dubbele-punt voor een LOGO-woord staat, weet het LOGO-systeem dat het om een naam van een variabele gaat en niet om de naam van een procedure of een LOGO-opdracht. In de bovenstaande LOGO-procedure zien we hoe de procedure zichzelf steeds met een nieuwe waarde voor ZIJDE aanroept. Ook dit programma vertalen we in BASIC. Hiermee kunt u elke denkbare rechtlijnige turtle-grafiek maken. U hoeft alleen de waarden van de invoerparameters (ZIJDE, GROEI en HOEK) te veranderen.

Het derde LOGO-programma tekent een reeks vierkanten in spiraalvorm. Het zijn bekende figuren in het 'land van de computergraphics'.

LOGO-programma nr. 3

```
TO VIERKANTSPIRAAL:ZIJDE:HOEK
IF:ZIJDE>150 THEN STOP
VIERKANT:ZIJDE  LEFT:HOEK
VIERKANTSPIRAAL:ZIJDE+4:HOEK
END
```

Ook dit LOGO-programma vertalen we in BASIC. Met deze drie programma's hebt u niet alleen met LOGO maar ook met de 'turtle-graphics' kennis gemaakt. Wilt u meer weten van deze graphics-wereld, dan wijzen wij op het boek *Turtle Geometry: The Computer as a Medium for Exploring Mathematics* van Harold Abelson en Andrea di Sessa, uitgegeven door Cambridge, MA:MIT Press, 1981. Liefhebbers van wiskunde, informatica en LOGO vinden in dit boek een schat aan informatie.

Natuurlijk zijn we in LOGO niet gebonden aan 'rechte lijnen'. In LOGO hebben we de beschikking over alle wiskundige functies en bewerkingen. Alle programma's uit dit boek kunnen in LOGO geschreven worden. Zij zouden dan vast korter, overzichtelijker en begrijpelijker geweest zijn. In LOGO kunnen we heel eenvoudig met lijsten en tabellen werken. In LOGO hoeven we de variabelen niet te declareren. Achter een naam in LOGO kan gewoon een getal, een string, een n-dimensionaal veld en nog veel meer schuil gaan. Ook kan het de naam van een procedure zijn. We hoeven ons niet te bekommeren om het reserveren van geheugenruimte (DIM-opdracht in BASIC), noch om het aangeven van het type (real, integer, character, boolean, enz.); het LOGO-systeem regelt dit zelf.

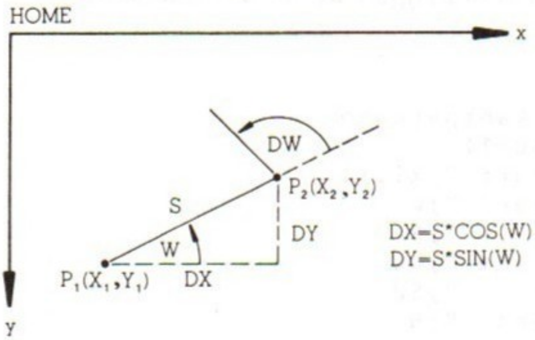
Wie in LOGO programmeert leert automatisch netjes programmeren. Begrippen als procedure, recursie, herhaling, toekenning worden op een natuurlijke manier aangeleerd. Met een paar opdrachten is een kind al in staat ingewikkelde figuren te tekenen. Met LOGO degraderen we de computer niet tot een programmeerbare zakrekenmachine, maar halen we er alles uit wat erin zit.

## Vertaling LOGO-programma nr. 1 in BASIC

De kern van elk LOGO-graphics-programma wordt gevormd door twee opdrachten:

| | |
|---|---|
| FORWARD (resp. BACK) | :ZIJDE |
| LEFT (resp. RIGHT) | :HOEK |

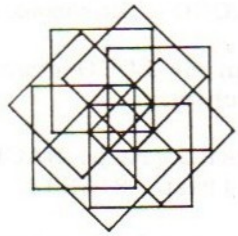
Hoe vertalen we deze opdrachten in BASIC? Bekijk hiertoe de volgende figuur.




De PEN staat in punt $P_1(x_1,y_1)$ en moet in de richting van hoek W over een afstand S verplaatst worden. In $P_2(x_2,y_2)$ aangekomen moet de PEN (eigenlijk de turtle) een hoek van DW° linksom maken. De volgende BASIC-opdrachten voeren dit uit:

```
INPUT "coordinaten start ";X1,Y1
INPUT "beginrichting(gr) ";W
PI=4*ATN(1)
H=.5:RD=PI/180:W1=W*RD
X2=INT(X1+S1*COS(W1)+H)
Y2=INT(Y1-S1*SIN(W1)+H)
LINE (FNX(X1),Y1)-(FNX(X2),Y2)
X1=X2:Y1=Y2
W=W+DW
IF W>360 THEN W=W-360
W1=W*RD
```

Deze opdrachten komt u in alle volgende BASIC-programma's tegen. Meer theorie is niet nodig! De programma's moeten met bovenstaande informatie gelezen kunnen worden.



Experimenteer met programma 31 LOGO-1 vierkantpatroon. Met W=90°, S1=20, DW=45, S2=75 en N=8 krijgen we de bovenstaande tekening.

```
100 REM programma 31 vierkantpatroon
                     (LOGO-1)
110 INPUT "coordinaten start ";X1,Y1
120 INPUT "beginrichting(gr) ";W
130 INPUT "verplaatsing      ";S1
140 INPUT "draaihoek linksom ";DW
150 INPUT "zijde vierkant    ";S2
160 INPUT "aantal vierkanen  ";N
170 PI=4*ATN(1)
180 H=.5:RD=PI/180:W1=W*RD
190 CLS
200 COLOR 1,15,4 : SCREEN 2
210 DEF FNX(X)=INT(37+X/1.4+.5)
220 FOR J=1 TO N
230  X2=INT(X1+S1*COS(W1)+H)
240  Y2=INT(Y1-S1*SIN(W1)+H)
250  LINE (FNX(X1),Y1)-(FNX(X2),Y2)
260  X1=X2:Y1=Y2
270  AX=X1:AY=Y1
280  W=W+DW
290  IF W>360 THEN W=W-360
300  W1=W*RD
310  FOR K=0 TO 2
320   X2=INT(X1+S2*COS(W1+K/2*PI)+H)
330   Y2=INT(Y1-S2*SIN(W1+K/2*PI)+H)
340   LINE (FNX(X1),Y1)-(FNX(X2),Y2)
350   X1=X2:Y1=Y2
360  NEXT K
370  LINE (FNX(X1),Y1)-(FNX(AX),AY)
380  X1=AX:Y1=AY
390 NEXT J
400 A$=INPUT$(1)
410 COLOR 15,4,4: END
```

W=90, S1=30, DW=48, S2=70 en N=15 geeft de onderstaande tekening. Als DW een deler is van 360, is N gelijk aan 360/DW. Kiest u voor DW een willekeurige waarde, neem dan voor N een groot getal, bijvoorbeeld 100, en breek het tekenen met de stoptoets af. De 'kinderen' doen dat in LOGO ook op deze manier.



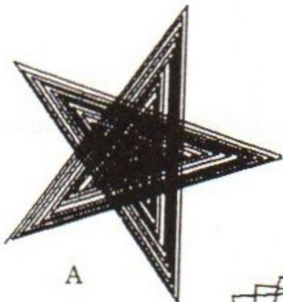
## Vertaling LOGO-programma nr.2 in BASIC

Het programma lijkt sterk op het vorige en behoeft daarom geen commentaar.

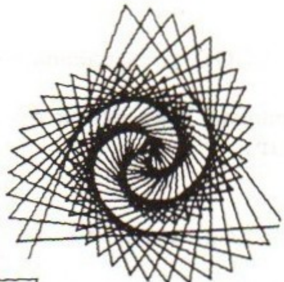
```
100 REM programma 32 turtle-grafiek
                         (LOGO-2)
110 INPUT "coordinaten start ";X1,Y1
120 INPUT "beginrichting(gr) ";W
130 INPUT "verplaatsing      ";S
140 INPUT "draaihoek linksom ";DW
150 INPUT "toename zijde     ";DS
160 PI=4*ATN(1)
170 H=.5:RD=PI/180:W1=W*RD
180 CLS
190 COLOR 1,15,4 : SCREEN 2
200 DEF FNX(X)=INT(37+X/1.4+.5)
210  X2=INT(X1+S*COS(W1)+H)
220  Y2=INT(Y1-S*SIN(W1)+H)
230  IF X2<0 OR X2>255 OR
        Y2<0 OR Y2>191
         THEN A$=INPUT$(1):STOP
240  LINE (FNX(X1),Y1)-(FNX(X2),Y2)
250  X1=X2:Y1=Y2
260  W=W+DW
270  IF W>360 THEN W=W-360
280  W1=W*RD:S=S+DS
290 GOTO 210
300 A$=INPUT$(1)
310 COLOR 15,4,4: END
```

Het is handig de ingevoerde waarden na het tekenen op het scherm of op een printer te laten afdrukken. In de volgende tabel zien we de diverse waarden voor de daaronderstaande tekeningen.

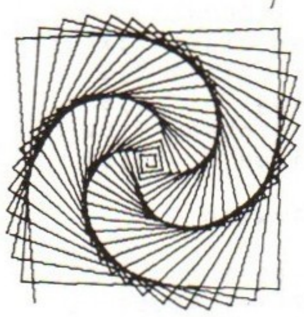
| Tekening | X1 | Y1 | W | S | DW | DS |
|---|---|---|---|---|---|---|
| A | 128 | 96 | 90 | 5 | 144 | 3 |
| B | 128 | 96 | 90 | 5 | 123 | 2 |
| C | 128 | 96 | 90 | -2 | 92 | 2 |
| D | 128 | 96 | 90 | 5 | 72 | 1 |
| E | 128 | 165 | 90 | 160 | 145 | 0 |



A



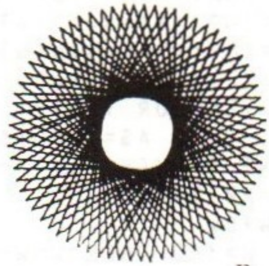
B



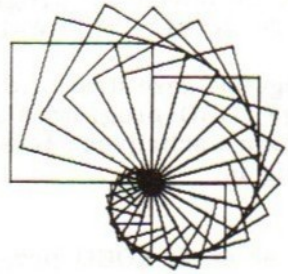
C



D



E



## Vertaling LOGO-programma nr.3 in BASIC

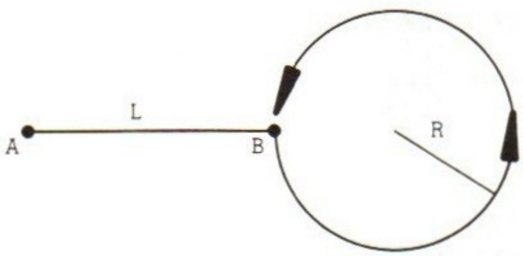
```
100 REM programma 33 vierkantspiraal
                        (LOGO-3)
110 INPUT "coordinaten start ";X1,Y1
120 INPUT "beginrichting(gr) ";W
130 INPUT "verplaatsing      ";S
140 INPUT "draaihoek linksom ";DW
150 INPUT "toename zijde     ";DS
160 PI=4*ATN(1)
170 H=.5:RD=PI/180:W1=W*RD
180 CLS
190 COLOR 1,15,4 : SCREEN 2
200 DEF FNX(X)=INT(37+X/1.4+.5)
210 AX=X1:AY=Y1
220   FOR FASE=0 TO PI STEP PI/2
230     GOSUB 1000 'punt (X2,Y2)
240     IF VLAG=1
          THEN A$=INPUT$(1):
               COLOR 15,4,4:STOP
250     LINE(FNX(X1),Y1)-(FNX(X2),Y2)
260     X1=X2:Y1=Y2
270   NEXT FASE
280   LINE (FNX(X1),Y1)-(FNX(AX),AY)
290   X1=AX:Y1=AY
300   W=W+DW:IF W>360 THEN W=W-360
310   W1=W*RD:S=S+DS
320 GOTO 220
330 END
1000 REM subroutine (X2,Y2)
1010  X2=INT(X1+S*COS(W1+FASE)+H)
1020  Y2=INT(Y1-S*SIN(W1+FASE)+H)
1030  IF X2<0 OR X2>255 OR
        Y2<0 OR Y2>191
         THEN VLAG=1
1040 RETURN
```

Voor de boven het programma staande tekening kozen we X1=128, Y1=96, W=90, S=5, DW=15 en DS=3,5.

Vergelijk deze drie BASIC-programma's eens met de overeenkomstige LOGO-programma's. In BASIC moeten we zelf alle graphicsoftware schrijven, terwijl deze in LOGO als machine-routines aanwezig is.

## Het 4e en 5e LOGO-programma

Tot slot van dit hoofdstuk geven we nog twee BASIC-programma's waarmee de turtle behalve rechte wegen ook kromme wegen kan bewandelen. Dit voegt een nieuw element aan de turtle-graphics toe. De LOGO-structuur zullen we stapsgewijs verfijnen. Als uitgangspunt nemen we de onderstaande figuur. Deze bestaat uit een lijnstuk L, met daaraan vast een cirkel met straal R. We noemen deze figuur voor het gemak even 'de steelpan'.




Als de turtle de steelpan tekent, begint hij in A. Hij legt vervolgens een afstand L in positieve x-richting af en komt in B. Hier draait hij 90° met de klok mee en legt daarna de omtrek van de cirkel af tot hij weer in B uitkomt. Daar draait hij zich 90° met de klok mee en kijkt dan rechtvooruit naar het punt A. Met deze steelpan kunnen we twee soorten tekeningen maken. Laten we deze mogelijkheden bekijken:

LOGO-programma 4

```
TO CIRKELFIGUUR-1:LIJNSTUK:STRAAL:HOEK
STEELPAN:LIJNSTUK:STRAAL
LEFT:HOEK
CIRKELFIGUUR-1:LIJNSTUK:STRAAL:HOEK
END
```

LOGO-programma 5

```
TO CIRKELFIGUUR-2:LIJNSTUK:STRAAL:HOEK
REPEAT 4 (STEELPAN:LIJNSTUK:STRAAL LEFT 90)
LEFT:HOEK
CIRKELFIGUUR-2:LIJNSTUK:STRAAL:HOEK
END
```

Deze twee LOGO-programma's kunnen nog niet uitgevoerd worden, want de procedure STEELPAN is nog niet in het LOGO-systeem gedefinieerd. We gaan de structuur verfijnen:

```
TO STEELPAN:LIJNSTUK:STRAAL
FORWARD:LIJNSTUK  RIGHT 90
CIRKEL:STRAAL
RIGHT 90
END
```

Ook in deze procedure zien we een nog niet gedefinieerde procedure, CIRKEL, opduiken. Om de turtle een cirkel te kunnen laten maken zullen we deze procedure CIRKEL moeten maken. Veel LOGO-versies bevatten reeds zo'n voorgedefinieerde CIRKEL-procedure. MSX-BASIC beschikt ook over een CIRCLE-functie. We zien deze in regel 290 van het volgende programma. Bedenk dat de X-coördinaat van het middelpunt met de FNX-functie moet worden aangeroepen, anders krijgen we een ellips.

```
CIRCLE(FNX(U),V),R,1,0,2*PI,1.4
```

- FNX(U),V: coördinaten middelpunt
- R: straal
- 1: afdrukkleur (1=zwart)
- 0: begin tekenhoek
- 2*PI: eind tekenhoek
- 1.4: correctiefactor voor de horizontale straal

Als we echt LOGO in MSX-BASIC willen nadoen, moeten de cirkels getekend worden vanuit het uiteinde van de steel (zie punt B in tekening op p.92). De MSX-CIRCLE-functie begint met tekenen aan de andere kant van de cirkel.

Als we het 'echt' willen doen moeten we de regels 290 t/m 320 vervangen door:

```
AX=X2 : AY=Y2
REM CIRKEL M(U,V) EN STRAAL R
U=INT(AX+R*COS(W1)+H)
V=INT(AY-R*SIN(W1)+H)
X1=AX : Y1=AY
FOR WW=(W1-PI) TO (W1+PI) STEP PI/32
:   X2=INT(U+R*COS(WW)+H)
:   Y2=INT(V-R*SIN(WW)+H)
:   LINE (FNX(X1),Y1)-(FNX(X2),Y2)
    X1=X2:Y1=Y2
NEXT WW
W=W+DW : IF W>=360 THEN W=W-360
W1=W*RD : X1=AX : Y1=AY
```

**Vertaling van LOGO-programma nr.4 in BASIC**



```
100 REM programma 34 cirkeliguur-1
                    (LOGO-4)
110 CLS
120 INPUT "coordinaten start ";X1,Y1
130 INPUT "beginrichting(gr) ";W
140 INPUT "lengte steel      ";L
150 INPUT "straal cirkel     ";R
160 INPUT "draaihoek linksom ";DW
170 PI=4*ATN(1)
180 H=.5:RD=PI/180:W1=W*RD
190 CLS
200 COLOR 1,15,4 : SCREEN 2
210 DEF FNX(X)=INT(37+X/1.4+.5)
220 XX=X1:YY=Y1
230 REM begin herhaal-lus
240   X2=INT(X1+L*COS(W1)+H)
250   Y2=INT(Y1-L*SIN(W1)+H)
260   LINE (FNX(X1),Y1)-(FNX(X2),Y2)
270   U=INT(X2+R*COS(W1)+H)
280   V=INT(Y2-R*SIN(W1)+H)
290   CIRCLE(FNX(U),V),R,1,0,2*PI,1.4
300   W=W+DW:IF W>=360 THEN W=W-360
310   W1=W*RD
320   X1=X2:Y1=Y2
330   IF X1<>XX OR Y1<>YY THEN 240
340 REM einde herhaal-lus
350 A$=INPUT$(1)
360 COLOR 15,4,4 : END
```

De bovenstaande tekening krijgen we door X1=128, Y1=135, W=90, L=90, R=20 en DW=170 te kiezen.

## Vertaling LOGO-programma nr.5 in BASIC

```
100 REM programma 35 cirkeliguur-2
                      (LOGO-5)
110 CLS
120 INPUT "coordinaten start ";X1,Y1
130 INPUT "beginrichting(gr) ";W
140 INPUT "lengte steel       ";L
150 INPUT "straal cirkel      ";R
160 INPUT "draaihoek linksom ";DW
170 PI=4*ATN(1)
180 H=.5:RD=PI/180:W1=W*RD
190 CLS
200 COLOR 1,15,4 : SCREEN 2
210 DEF FNX(X)=INT(37+X/1.4+.5)
220 REM begin eindloze lus
230 FOR J=1 TO 4
240   X2=INT(X1+L*COS(W1)+H)
250   Y2=INT(Y1-L*SIN(W1)+H)
260   LINE (FNX(X1),Y1)-(FNX(X2),Y2)
270   REM cirkel met middelpunt
          (U,V) en straal R
280   U=INT(X2+R*COS(W1)+H)
290   V=INT(Y2-R*SIN(W1)+H)
300   CIRCLE(FNX(U),V),R,1,0,2*PI,1.4
310   W=W+90 : IF W>=360 THEN W=W-360
320   W1=W*RD
330   X1=X2:Y1=Y2
340 NEXT J
350 W=W+DW
360 W1=W*RD
370 GOTO 230
380 REM 'einde' eindloze lus
390 END
```



Voor deze tekening geldt: X1=128, Y1=96, W=0, L=50, R=10 en DW=45.
Met andere waarden voor deze variabelen kunt u de mooiste turtleplaatjes maken.

# 7 *Educatieve toepassingsprogramma's*

De voorgaande 35 grafische programma's zijn geen toepassingsprogramma's. Ze waren bedoeld om een overzicht te geven van de mogelijkheden van graphics met hoog oplossend vermogen. In dit hoofdstuk zullen we vijf praktijkgerichte grafische programma's presenteren. Deze programma's zijn:

1. Tekenen van een landkaart
2. Maken van een histogram (een stavengrafiek)
3. Demonstratieprogramma voor de breking van lichtstralen
4. Demonstratieprogramma voor de 'speldenworp' van Buffon
5. Prooi-roofdierpopulaties

## 1. Tekenen van eén landkaart

Educatieve programma's over topografie gebruiken dikwijls landkaarten van een land of van een werelddeel die door de computer getekend worden. We zullen laten zien hoe de computer zo'n landkaart kan tekenen. We gaan de landkaart (althans de grensomtrek) van Zwitserland tekenen. Waarom Zwitserland? Wel, in de eerste plaats omdat de auteur dit gekozen heeft en in de tweede plaats omdat Zwitserland geen eilanden heeft. Het tekenen van de contouren van Nederland gaat in principe net zo, alleen kosten al die eilanden een hoop extra werk, vandaar!

Hoe tekenen we nu de omtrek van een bepaald land? Heel eenvoudig! We pakken gewoon een atlas, een stukje overtrekpapier, een potlood, een lineaal en millimeterpapier. We kiezen in de atlas een land of werelddeel uit en trekken het op overtrekpapier over. Daarna leggen we het overgetrokken plaatje op millimeterpapier, waar we van te voren een coördinatenstelsel op getekend hebben (een x- en een y-as). We bepalen nu van een (groot) aantal punten op de omtrek van het land (of werelddeel) de coördinaten (x,y) en schrijven deze op. Als we ons op het beeldscherm ook een coördinatenstelsel voorstellen en als we hierin de punten, waarvan we de coör-

dinaten in een programma opnemen, met elkaar laten verbinden, dan ontstaat een 'landkaart' op het scherm.

In het onderstaande programma, dat 'de kaart' van Zwitserland tekent, is een 'echte' kaart gebruikt met een schaal van 1 : 2.000.000. Om een enigszins natuurgetrouwe weergave te verkrijgen zijn de coördinaten van 90 grenspunten berekend. De coördinaten van deze punten, die opgegeven zijn in millimeters ten opzichte van de oorsprong van het gekozen coördinatenstelsel, zijn in DATA-regels opgenomen. Als het programma het coördinatenpaar (0,0) leest, weet het dat de 'kaart' af is.

De x-coördinaten liggen tussen 6 en 177. Om deze naar het hoge-resolutiebereik 0-255 en 0-191 te transformeren vermenigvuldigen wij de x- en y-coördinaat met 1,35. Hierdoor blijft er nog ruimte over voor bijvoorbeeld een kader rond de kaart. De werking van het programma zal hiermee duidelijk zijn.

We hebben op deze manier ook een kaart van Europa en zelfs een wereldkaart getekend. Het probleem met de eilanden hebben we als volgt opgelost. Als het programma in een DATA-regel negatieve x- en y-coördinaten leest, weet het programma dat dit punt niet met het vorige verbonden moet worden en dat dit punt dus het begin is van een apart stukje 'land'.



```
100 REM programma 36 kaart
                     van zwitserland
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 K=1.35:H=.5:U=18:V=191
150 READ X,Y
160 X1=INT(U+K*X+H)
170 Y1=INT(V-K*Y+H)
180 REM begin herhaal lus
190    X2=INT(U+K*X+H)
200    Y2=INT(V-K*Y+H)
210    LINE (FNX(X1),Y1)-(FNX(X2),Y2)
220    X1=X2:Y1=Y2
230    READ X,Y
240    IF X<> 0 THEN 190
250 REM einde herhaal lus
260 A$=INPUT$(1)
270 COLOR 15,4,4 : END
280 '
290 DATA 69,108,71,107,70,104,75,104
300 DATA 75,104,76,106,80,107,81,104
310 DATA 86,105,91,108,94,107,101,106
320 DATA 100,108,105,108,106,110,101
330 DATA 110,98,112,102,117,108,118
340 DATA 112,112,114,115,118,110,128
350 DATA 110,139,102,145,103,146,95
360 DATA 142,86,144,78,154,77,154,77
370 DATA 154,72,163,67,168,68,173,76
380 DATA 177,73,174,59,177,56,177,52
390 DATA 171,51,167,56,161,50,165,43
400 DATA 166,34,162,34,157,42,143,36
410 DATA 139,48,136,45,133,48,132,40
420 DATA 122,23,125,15,122,11,119,12
430 DATA 114,20,116,25,100,35,102,45
440 DATA 94,42,88,35,90,28,79,15,75
450 DATA 15,66,19,60,14,52,12,48,13
460 DATA 37,29,39,36,37,40,39,43,29
470 DATA 45,16,38,18,33,13,29,6,28,6
480 DATA 32,11,34,13,40,10,45,12,53
490 DATA 25,63,26,73,30,73,48,94,42
500 DATA 94,46,102,54,102,53,99,61
510 DATA 98,63,102,69,108
520 DATA 0,0
```

Zie de appendix voor een uitbreiding van dit programma.

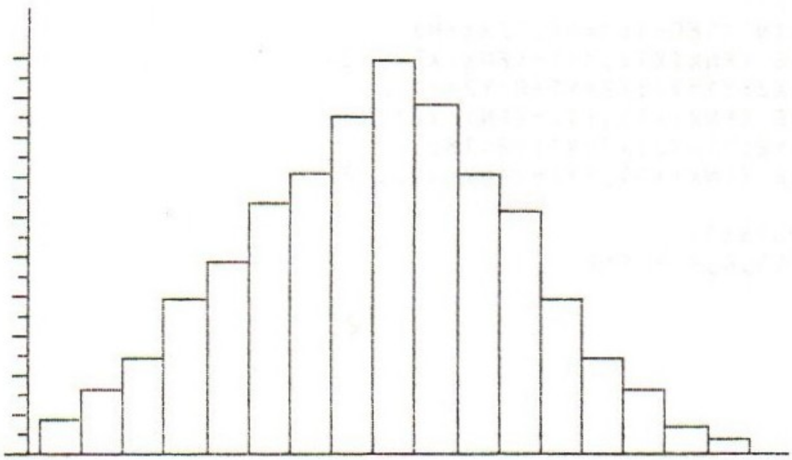
## 2. Maken van een histogram

Vaak willen we een aantal waarnemingen grafisch in een stavendiagram of histogram weergeven. Hiertoe dient het volgende programma. Dit programma tekent een horizontale en een verticale as. De verticale as wordt in stukjes van acht schermpuntjes verdeeld. Dit komt overeen met 5% van de hele verticale as. Elk tweede streepje op deze as geeft de volgende 10% aan en is iets breder getekend. Hiermee kan de lengte van de staven redelijk geschat worden.

Het programma kan maximaal 100 waarnemingen verwerken. Uit estetische overwegingen moet u echter niet veel meer dan 40 waarnemingen invoeren, omdat de staafjes anders meer op lijntjes dan op balkjes gaan lijken.

De waarnemingen worden zo 'geschaald' dat de grootste waarneming precies met 160, de lengte van de verticale schaalverdeling, overeenkomt.

Het zou mooi zijn als we bij de staven, de assen en de schaalverdeling tekst en getallen konden zetten, maar dat gaat heel lastig, dus doen we het nu maar niet. Denkt u erom dat de getallen die u intoetst de lengte van de staven aangeven. Als elke staaf een bepaalde klasse met waarnemingen voorstelt, dan voert u dus steeds het aantal waarnemingen (de frequentie) van een klasse in. Het programma kan uitgebreid worden door er een stuk voor te zetten dat ruwe gegevens inleest, die vervolgens netjes in klassen verdeeld worden. De frequentie van de waarnemingen in de klassen wordt vervolgens gebruikt om de stavengrafiek te tekenen.

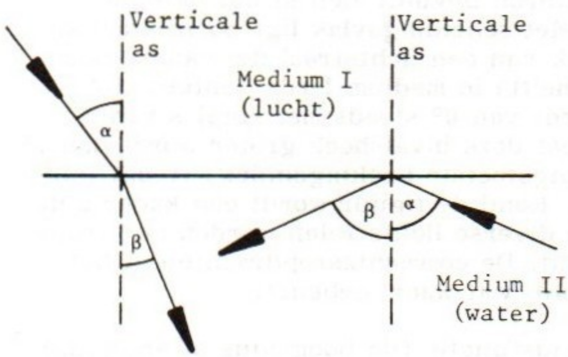


```
100 REM programma 37 histogram
110 CLS
120 PRINT "histogram tekenen"
130 PRINT "-----------------"
140 PRINT:PRINT
150 INPUT "hoeveel gegevens(<40)";N
160 DIM A(N)
170 MX=-1000000000! : PRINT
180 FOR J=1 TO N
190     PRINT "waarde"; J; TAB(12);
200     INPUT A(J)
210     IF A(J)>MX THEN MX=A(J)
220 NEXT J
230 CLS
240 COLOR 1,15,4 : SCREEN 2
250 DEF FNX(X)=INT(37+X/1.4+.5)
260 REM horizontale as
270 LINE (FNX(0),180)-(FNX(256),180)
280 REM verticale as
290 LINE (FNX(10),0)-(FNX(10),180)
300 REM schaalverdeling
310 FOR J=10 TO 1 STEP -1
320     X1=7:Y1=180-J*16:X2=10:Y2=Y1
330     LINE (FNX(X1),Y1)-(FNX(2),Y2)
340     X1=4:Y1=Y2+8:X2=10:Y2=Y1
350     LINE (FNX(X1),Y1)-(FNX(2),Y2)
360 NEXT J
370 REM staven tekenen (B=breedte)
380 B=INT(240/N)
390 FOR J=1 TO N
400     X1=(J-1)*B+15:Y1=180
410     X2=X1:
        Y2=INT(180-160*A(J)/MX+H)
420     LINE (FNX(X1),Y1)-(FNX(X2),Y2)
430     X1=X2:Y1=Y2:X2=X1+B:Y2=Y1
440     LINE (FNX(X1),Y1)-(FNX(X2),Y2)
450     X1=X2:Y1=Y2:X2=X1:Y2=180
460     LINE (FNX(X1),Y1)-(FNX(X2),Y2)
470 NEXT
480 A$=INPUT$(1)
490 COLOR 15,4,4 : END
```

### 3. Demonstratieprogramma voor de breking van lichtstralen

Met dit programma willen we laten zien hoe de MSX-computer bij natuurkundelessen gebruikt zou kunnen worden. Voordat we het programma geven leggen we nog iets uit van de natuurkundige beginselen van de breking van licht.

Als een lichtstraal vanuit de ene stof (bijvoorbeeld lucht) een andere, optisch dichtere, stof (bijvoorbeeld water) binnenkomt, wordt de straal op het scheidingsvlak van beide stoffen naar de verticale as toe gebogen. Zie onderstaande figuur.




Hierbij geldt de brekingswet van Snellius:

$$\frac{\sin\alpha}{\sin\beta} = \frac{c_1}{c_2} = n$$

$c_1$ en $c_2$ zijn de lichtsnelheden in respectievelijk de eerste en de tweede stof. De constante n heet de brekingsindex. Bij de overgang van lucht naar water geldt een brekingsindex van 1,33. Voor de overgang van lucht naar glas gelden andere waarden, enzovoorts.

Als het licht vanuit een 'dichter' medium overgaat in een 'dunner' medium geldt het omgekeerde: de lichtstralen worden nu van de verticale as, die loodrecht op het scheidingsvlak van beide stoffen staat, afgebogen. De brekingshoek α kan nooit groter dan 90° zijn.

Voor dit grensgeval ($\alpha = 90°$) geldt:

$$\frac{\sin 90°}{\sin\beta^*} = n \Rightarrow \sin\beta^* = \frac{1}{n} \quad (\text{want } \sin 90° = 1)$$

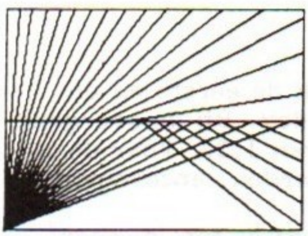
Bij de overgang van lucht naar water geldt voor β*:

$$\sin\beta^* = \frac{1}{1,33} \Rightarrow \sin\beta^* = 0,7519 \Rightarrow \beta^* = 48,75°$$

Groter dan 48,75° kan β dus niet worden. Dit betekent dat als het licht van water overgaat in lucht met een invalshoek α die groter is dan β* (zie rechter figuur), het licht niet meer het water 'uitkomt'. Op het scheidingsvlak van water en lucht wordt het licht geheel teruggekaatst. Op dit principe berusten de moderne glasvezelkabels, waarin informatie in de vorm van licht wordt getransporteerd.

In het onderstaande demonstratieprogramma kan de brekingsindex n ingetoetst worden. De lichtbron bevindt zich in het punt met schermcoördinaten (0,190). Het scheidingsvlak ligt horizontaal en is de lijn V=96. De invalshoek van een lichtstraal die van medium 2 (de dichtere stof, onderste helft) in medium 1 (de lichtere stof, bovenste helft) overgaat wordt van 0° steeds met stapjes van 3° opgehoogd. Op het moment dat deze invalshoek groter wordt dan β* (deze is afhankelijk van de ingetoetste brekingsindex) treedt totale reflectie (terugkaatsing) op. Rond de figuur wordt een kader getekend. De eindpunten van de diverse lichtstralen worden met trigonometrische functies berekend. De commentaaropdrachten in het programma leggen nog eens uit 'wat waar' gebeurt.

De inverse functie van de sinusfunctie (de boogsinus of arcsinus) bestaat niet in MSX-BASIC. We lossen dit op door de inverse tangensfunctie (ATN in BASIC) te gebruiken.

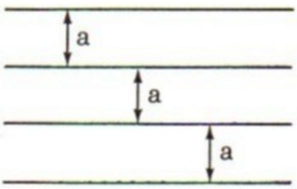


Het voordeel van een dergelijke computersimulatie, boven het uitvoeren van het natuurkundige experiment, is dat we heel gemakkelijk verschillende brekingsindexen kunnen invoeren zonder steeds andere apparatuur op te hoeven stellen en dat de lichtstralen als dunne lijnen zichtbaar gemaakt kunnen worden.

```
100 REM programma 38 breking van
                          licht
110 CLS
120 INPUT "brekingsindex N";N
130 PI=4*ATN(1)
140 V=96:H=.5:RD=PI/180
150 CLS
160 COLOR 15,4,4 : SCREEN 2
170 DEF FNX(X)=INT(37+X/1.4+.5)
180 LINE (FNX(0),V)-(FNX(255),V)
190 LINE (FNX(0),190)-(FNX(255),190)
200 LINE (FNX(255),190)-(FNX(255),0)
210 LINE (FNX(255),0)-(FNX(0),0)
220 LINE (FNX(0),0)-(FNX(0),190)
230 REM stralen in medium 1 en 2
        tekenen;B=beta in graden;
        B1=beta in radialen; A1=
        alfa i rad.; SA=sin(alfa)
240 B=B+3:B1=B*RD:X1=0:Y1=190
250 X2=INT(90*TAN(B1)+H):Y2=V
260 IF X2>255
      THEN A$=INPUT$(1):
           COLOR 15,4,4 : STOP
270 LINE (FNX(X1),Y1)-(FNX(X2),Y2)
280 X1=X2:Y1=Y2
290 REM alfa en sin(alfa) berekenen
        en totale reflectie nagaan
300 S=N*SIN(B1)
310 IF S>1 THEN GOSUB 1000
           ELSE GOSUB 2000
320 GOTO 240
330 END
1000 REM subroutine totale reflectie
1010  X2=X1+X1:Y2=190
1020  IF X2<255
        THEN LINE (FNX(X1),Y1)-
                  (FNX(X2),Y2)
        ELSE X2=255:Y2=INT(V+(255-X1)
                  /TAN(B1)+H):
             LINE (FNX(X1),Y1)-
                  (FNX(X2),Y2)
1030 RETURN
2000 REM subroutine breking
2010  A1=ATN(S/SQR(1-S*S))
2020  X2=INT(X1+90*TAN(A1)+H):Y2=0
2030  IF X2<255
        THEN LINE (FNX(X1),Y1)-
                  (FNX(X2),Y2)
        ELSE X2=255:Y2=INT(V-(255-X1)
                  /TAN(A1)+H):
             LINE (FNX(X1),Y1)-
                  (FNX(X2),Y2)
2040 RETURN
```

## 4. De speldenworp van Buffon (1773)

Stelt u zich voor dat we in een plat vlak een aantal evenwijdige lijnen met een onderlinge afstand a tekenen.




We werpen nu, zonder echt te 'mikken', een speld met een lengte c die kleiner dan of gelijk aan a is ($c \leq a$) op het vlak met de evenwijdige lijnen. Hoe groot is nu de kans dat een speld een van de lijnen treft, dat wil zeggen snijdt of raakt?



speld snijdt een lijn



speld raakt een lijn

Dit probleem kwam in 1773 op bij de Franse wiskundige Buffon na het zien van de Amerikaanse vlag met de 'stars en stripes'. Buffon heeft berekend dat deze kans gelijk is aan

$$\frac{2c}{a\pi}$$

Omdat deze formule de constante $\pi$ bevat, heeft men deze 'speldenworp' vaak gebruikt om de waarde van $\pi$ door simulatie te bepalen. We werpen hiertoe vaak, bijvoorbeeld een lucifer, op een papier waarop een aantal evenwijdige lijnen (met een onderlinge afstand die groter dan of gelijk aan de lengte van de lucifer is). Als de lucifer in n worpen k keer een lijn treft, dan geldt dat

$\frac{2c}{a\pi}$ ongeveer gelijk is aan $\frac{k}{n}$.

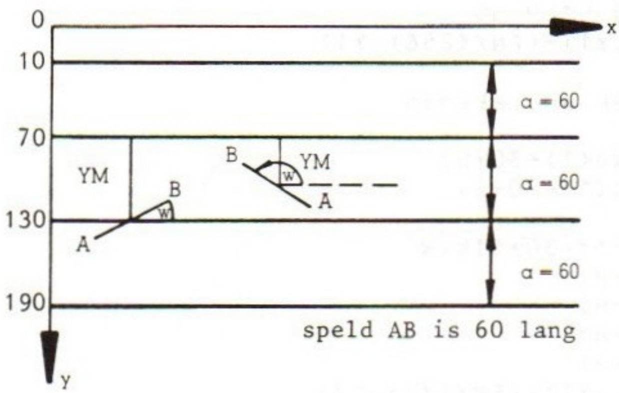
Beide uitdrukkingen geven namelijk een indruk van de kans dat de lucifer een lijn treft.
We kunnen als volgt de waarde van $\pi$ benaderen:

$$\frac{2c}{a\pi} = \frac{k}{n} \Rightarrow \pi \simeq \frac{2cn}{ak}.$$

Op deze manier heeft de astronoom Rudolf Wolf in 1850 met 5000 luciferworpen voor π de waarde 3,1596 gevonden (π = 3,141592654....).

Met een computer kunnen we gemakkelijk duizenden worpen simuleren. Deze programma's vinden we in bijna elk informaticaleerboek. Dit zijn echter programma's die vrij lang draaien, maar waaraan niets te 'zien' is. Als we bijvoorbeeld in zo'n programma voor n de waarde 10.000 zouden intikken, zien we eerst een tijd niets en vervolgens verschijnt de mededeling dat van de 10.000 keer 6.349 keer een lijn is geraakt, hetgeen voor π de waarde 3,1501024 oplevert.

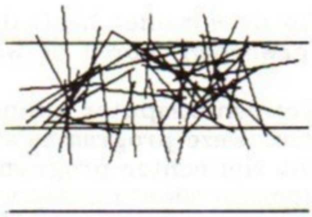
We gaan een programma maken dat ook dergelijke berekeningen uitvoert, maar dat bovendien elke speld die geworpen wordt laat zien. We zien het experiment als een film aan ons voorbijgaan. Bekijk hiertoe de onderstaande tekening.




De manier waarop een speld valt kan met twee toevalsgetallen bepaald worden. De twee toevalsgetallen bepalen respectievelijk de afstand YM van de speld tot de daarboven liggende lijn en de hoek W die de speld met de lijnen maakt:

| | | |
|---|---|---|
| $0 \leq YM \leq a$ | ; | YM is de afstand van het midden van de speld tot de daarboven liggende horizontale lijn |
| $0 \leq W \leq 180°$ | ; | W is de hoek (in positieve zin, dat wil zeggen bij draaiing tegen de klok in) die de speld met de positieve x-richting maakt |

Hierna volgt een illustratie van hoe het eruit kan zien en het programma.



```
100 REM programma 39 speldenworp van
                      buffon
110 CLS
120 PRINT "speldenworp van Buffon"
130 PRINT "----------------------"
140 PRINT:PRINT
150 INPUT "hoeveel worpen";N
160 M=0:H=.5:PI=4*ATN(1)
170 CLS
180 COLOR 1,15,4 : SCREEN 2
190 DEF FNX(X)=INT(37+X/1.4+.5)
200 REM lijnen tekenen
210 FOR Y1=10 TO 190 STEP 60
220    LINE (FNX(0),Y1)-(FNX(256),Y1)
230 NEXT Y1
240 REM n maal gooien en tekenen
250 FOR J =1 TO N
260    XM=INT(226*RND(1)+30+H)
270    YM=INT(60*RND(1)+70+H)
280    W=PI*RND(1)
290    DX=30*COS(W):DY=30*SIN(W)
300    X1=INT(XM-DX+H)
310    Y1=INT(YM+DY+H)
320    X2=INT(XM+DX+H)
330    Y2=INT(YM-DY+H)
340    LINE (FNX(X1),Y1)-(FNX(X2),Y2)
350    IF Y1>130 OR Y2<=70
         THEN M=M+1
360 NEXT J
370 A$=INPUT$(1)
380 COLOR 15,4,4 : SCREEN 0
390 CLS
400 PRINT "aantal worpen         ";N
410 PRINT "aantal doorsnijdingen";M
420 PRINT "benadering voor pi";
          INT(2*N/M*10000+.5)/10000
430 END
```

Kijk in de appendix hoe we de tekst en de waarden uit de regels 400, 410 en 420 onder de tekening op het graphic-scherm kunnen afdrukken.

Uit de waarden voor YM en W berekenen we de coördinaten van de uiteinden A en B van de speld. Het programma tekent hiermee de speld op het beeldscherm.

Een speld treft een lijn als de y-coördinaat van A groter dan of gelijk aan 130 is of als de y-coördinaat van B kleiner dan of gelijk aan 70 is. De x-coördinaten van A en B doen er in het geheel niet toe.

Het programma tekent als 'speelveld' vier evenwijdige lijnen met een onderlinge afstand van 60. De spelden zijn trouwens ook 60 lang. Als alle spelden geworpen zijn kunnen door een toets in te drukken de waarden voor n, k en $\pi$ worden afgelezen.

## 5. Prooi-roofdierpopulaties

Het laatste educatieve programma is een (deterministische) simulatie van een ecologisch systeem. Het is een demonstratieprogramma voor een biologieles.

Het ecologische systeem bevat gras, hazen en vossen. Tussen deze drie ecologische componenten gelden de volgende betrekkingen:

1. De hazen eten gras en de vossen eten hazen.
2. Als er meer gras groeit, neemt ook het aantal hazen toe. Deze hazen eten echter van het gras en verminderen zo hun eigen groei.
3. Als er meer hazen komen, neemt ook het aantal vossen toe. Omdat vossen hazen eten, verminderen zij zelf hun groei, net zoals bij de hazen en het gras.

Als we het aantal hazen op een bepaald tijdstip t aangeven met h(t) en het aantal vossen met v(t), kunnen we, volgens Lotka en Volterra (1920), voor het aantal hazen en vossen op tijdstip t+1 de volgende vergelijkingen opstellen:

$$h(t+1) = h(t) + a.h(t) - b.h(t).v(t) \qquad \text{vergelijking (1)}$$
$$v(t+1) = v(t) + c.v(t).h(t) - d*v(t) \qquad \text{vergelijking (2)}$$

De toename van het aantal hazen tussen de tijdstippen t en t+1 is evenredig met het aantal hazen op tijdstip t, dus

$$\underbrace{h(t+1)-h(t)}_{\text{toename hazen}} = \underbrace{a}_{\text{groeifactor}}.\underbrace{h(t)}_{\text{aantal hazen op tijdstip t}}$$

De afname van het aantal hazen is evenredig met het aantal aanwezige hazen (natuurlijk verloop) en met het aantal vossen, want die eten hazen, dus

$$\underbrace{h(t+1)-h(t)}_{\text{groei van het aantal hazen}} = \underbrace{a.h(t)}_{\text{toename aantal hazen}} - \underbrace{b.h(t).v(t)}_{\text{afname aantal hazen}}$$

Dit is vergelijking (1). We nemen in het programma aan dat er altijd genoeg gras voor de hazen voorhanden is.

De toename van het aantal vossen is evenredig met het aantal aanwezige vossen en met het aantal hazen (prooi). De afname van het aantal vossen is alleen evenredig met het aantal, omdat in dit systeem de vossen zelf geen prooidieren zijn. Voor de vossen krijgen we dus:

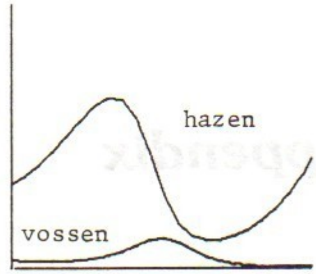
$$\underbrace{v(t+1)-v(t)}_{\text{groei van het aantal vossen}} = \underbrace{c.v(t).h(t)}_{\text{toename aantal vossen}} - \underbrace{d.v(t)}_{\text{afname aantal vossen}}$$

De vergelijkingen zijn als zogeheten differentievergelijkingen opgesteld. Het gras speelt, zoals gezegd, eigenlijk geen rol; er is altijd genoeg om alle hazen te voeden.

Bij het draaien van het simulatieprogramma hebben we de volgende waarden gekozen:
X (= h(0)) = 200; Y (= v(0)) = 20; a=0,3; b=0,01; c=0,002 en d=0,5.

De grafiek die het programma tekent laat heel mooi de groei en de afname van de hazen- en vossenpopulaties zien. Het aantal hazen groeit eerst, bereikt een maximum en neemt vervolgens af. De vossen groeien ook, maar later, bereiken later een maximum en nemen dan ook af, waarna de cyclus zich herhaalt. In de biologie noemen we dit een dynamisch evenwicht.

U hoeft de waarden voor de coëfficiënten a, b, c en d maar iets te veranderen of het systeem kan ontregeld worden. Hierbij neemt òf het aantal hazen enorm toe òf alle hazen en vossen sterven snel uit. Deze eenvoudige simulatie toont aan hoe desastreus een kleine ingreep in een bestaand ecologisch systeem dat in een dynamisch evenwicht is kan zijn.




```
100 REM progamma 40 roofdier-prooi-
                    dier ecosysteem
110 CLS
120 INPUT"pop. prooidieren(200)";X
130 INPUT"pop. roofdieren(20)";Y
140 INPUT"groei prooidieren(.3)";A
150 INPUT"afname prooidieren(.01)";B
160 INPUT"groei roofdieren(.002)";C
170 INPUT"afname roofdieren(.5)";D
180 CLS
190 COLOR 1,15,4 : SCREEN 2
200 DEF FNX(X)=INT(37+X/1.4+.5)
210 K=.3:H=.5:V=191
220 FOR X1=0 TO 255 STEP 8
230    XP=X+(A*X-B*X*Y)
240    YR=Y+(C*X*Y-D*Y)
250    REM pop.prooidieren tekenen
260    Y1=INT(V-K*X+H)
270    X2=X1+8:Y2=INT(V-K*XP+H)
280    IF Y2<0 OR Y2>191
         THEN A$=INPUT$(1): STOP
290    LINE (FNX(X1),Y1)-(FNX(X2),Y2)
300    REM pop.roofdieren
310    Y1=INT(V-K*Y+H)
320    X2=X1+8:Y2=INT(V-K*YR+H)
330    IF Y2<0 OR Y2>191
         THEN A$=INPUT$(1): STOP
340    LINE (FNX(X1),Y1)-(FNX(X2),Y2)
350    X=XP:Y=YR
360 NEXT X1
370 A$=INPUT$(1)
380 COLOR 15,4,4 : END
```

# Appendix

In deze appendix laten we zien hoe een aantal programma's uitgebreid kunnen worden om leuke effecten te bereiken. Ook geven we nog een aantal tekeningen die u met de programma's kunt maken. Tot slot volgt dan nog een betere versie van het bolprogramma 25.

## Programma 4

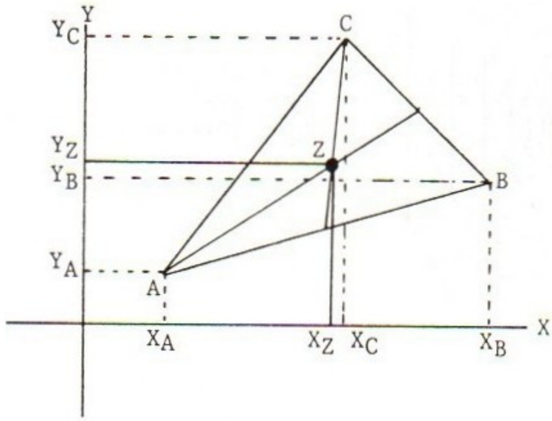
In de tekening, die programma 4 oplevert, zien we dat tussen een zeshoek en de daarbinnenliggende zeshoek zes driehoeken ontstaan. We gaan deze driehoeken om en om 'inkleuren'. We doen dit niet alleen voor de driehoeken tussen de buitenste zeshoek en de eerste ingeschreven zeshoek, maar ook bij de driehoeken tussen de eerste ingeschreven zeshoek en de tweede ingeschreven zeshoek, voor de driehoeken tussen de tweede ingeschreven zeshoek en de derde ingeschreven zeshoek, enzovoorts. Er ontstaan dan drie 'zwarte' spiraalvlakken.

Als we een vlak, in dit geval een driehoek, dat geheel omsloten wordt door lijnen willen inkleuren, moeten we in dat vlak een punt lokaliseren. De schermcoördinaten van dat punt gebruiken we dan in de PAINT-opdracht om het vlak waarin dat punt ligt te kleuren (zwart, of een andere kleur). Omdat er een groot aantal driehoeken gekleurd moeten worden, moeten we ook een groot aantal coördinaten van punten in die driehoeken berekenen. Deze punten moeten op een systematische manier in de FOR-NEXT-lus (regels 230-350) bepaald worden. Als we in het X-Y-vlak een driehoek ABC tekenen, waarvan de coördinaten van de hoekpunten $(X_A, Y_A)$; $(X_B, Y_B)$ en $(X_C, Y_C)$ zijn (zie onderstaande tekening), dan weten we uit de meetkunde dat de coördinaten van het zwaartepunt (Z) van de driehoek gelijk zijn aan

$$\left(\frac{X_A + X_B + X_C}{3} \;,\; \frac{Y_A + Y_B + Y_C}{3}\right).$$

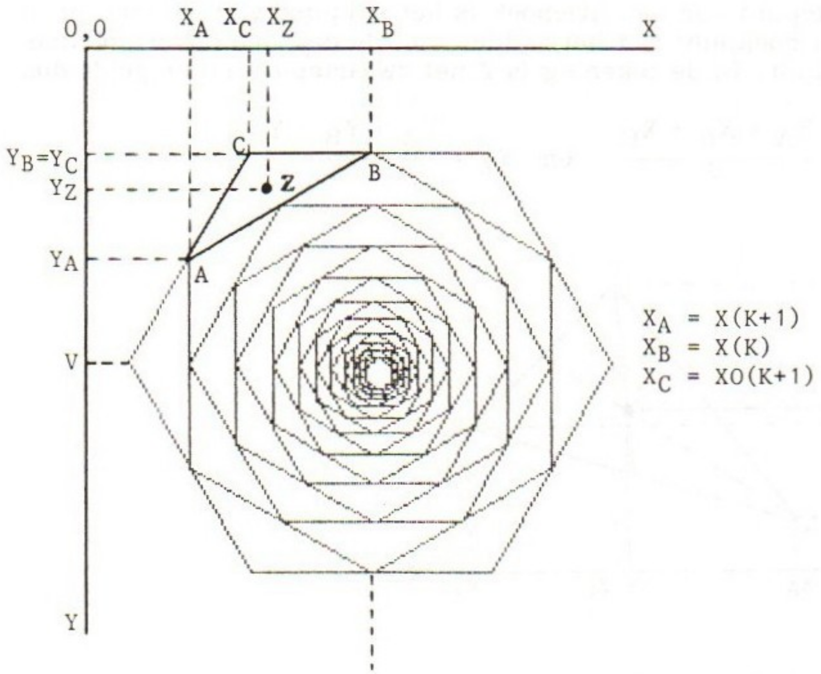
Het zwaartepunt van een driehoek is het snijpunt van de drie lijnen die elk een hoekpunt met het midden van de daartegenoverliggende zijde verbindt. In de tekening is Z het zwaartepunt en er geldt dus dat

$$X_Z = \frac{X_A + X_B + X_C}{3} \quad \text{en} \quad Y_Z = \frac{Y_A + Y_B + Y_C}{3} .$$



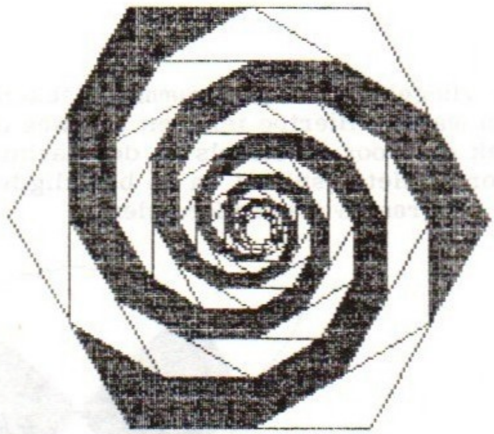

$X_Z = (X_A+X_B+X_C)/3$
$Y_Z = (Y_A+Y_B+Y_C)/3$

We kiezen in ons programma steeds het zwaartepunt van de driehoek als inwendig punt. Om de coördinaten hiervan de kunnen berekenen moeten we dus de drie x- en de drie y-coördinaten van de hoekpunten weten. We hebben in de onderstaande tekening, die door programma 4 getekend wordt, één driehoekje als voorbeeld genomen. Van dit driehoekje zijn de hoekpunten A en B twee hoekpunten van de ingeschreven zeshoek, terwijl hoekpunt C een hoekpunt is van de daarvoor getekende (in ons voorbeeld de buitenste) zeshoek. We moeten dus steeds van twee zeshoeken de coördinaten van alle zes hoekpunten ter beschikking hebben. In het onderstaande programma hebben we, om de coördinaten van de hoekpunten van de zojuist getekende zeshoek te kunnen onthouden voordat de nieuwe berekend worden, twee arrays extra opgenomen, en wel XO(6) en YO(6) (van XOud en YOud). In regel 325 maken we XO(J) en YO(J) gelijk aan de coördinaten van de zojuist getekende zeshoek (X(J) en Y(J)), vlak voordat X(J) en Y(J) gelijk worden gemaakt aan de coördinaten van de hoekpunten van de volgende zeshoek (MX(J) en MY(J)) in regel 330).




We zien vervolgens in de regels 263 en 264 hoe we de schermcoördinaten van het zwaartepunt berekenen uit de twee hoekpunten van de nieuwe zeshoek (X(K), X(K+1) en Y(K) en Y(K+1) en één hoekpunt van de daarvoor getekende zeshoek (XO(K+1) en YO(K+1)). In de FOR-NEXT-lus van regel 262 nemen we de stapgrootte 2 om steeds één driehoek over te slaan. Natuurlijk hoeven we dit inkleuren pas te doen als de tweede zeshoek getekend is, vandaar de IF N=1 THEN 270 in regel 261. Het inkleuren gebeurt tenslotte in regel 265 met PAINT(FNX(X),Y),1. Denk erom dat ook hier FNX(X) in plaats van X genomen wordt anders wordt een verkeerd vlak gekleurd.

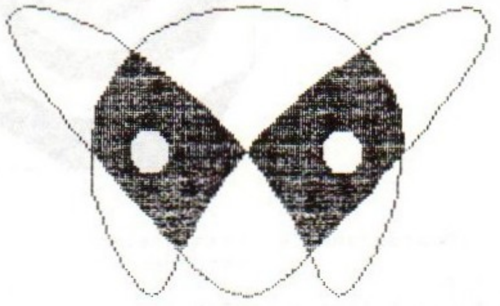
Het resultaat staat op p.113. Daarna volgt het aangepaste programma 4.



```
100 REM programma 4 ingeschreven
                    zeshoeken
110 CLS
120 COLOR 1,15,4 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 DIM X(6),Y(6),MX(6),MY(6),XO(6),
       YO(6)
150 PI=4*ATN(1)
160 U=128:V=96:R=100:H=.5
170 W=60*PI/180
180 FOR J=0 TO 6
190     W1=J*W
200     X(J)=INT(U+R*COS(W1)+H)
210     Y(J)=INT(V-R*SIN(W1)+H)
220 NEXT J
230 FOR N=1 TO 20
240    FOR J=0 TO 5
250      LINE (FNX(X(J)),Y(J))-
              (FNX(X(J+1)),Y(J+1))
260    NEXT J
261    IF N=1 THEN 270
262    FOR K=1 TO 5 STEP 2
263     X=INT((X(K)+X(K+1)+XO(K+1))
                                 /3)
264     Y=INT((Y(K)+Y(K+1)+YO(K+1))
                                 /3)
265     PAINT(FNX(X),Y),1,1
266    NEXT K
270    FOR K=0 TO 5
280     MX(K)=INT((X(K)+X(K+1))/2+H)
290     MY(K)=INT((Y(K)+Y(K+1))/2+H)
300    NEXT K
310    MX(6)=MX(0) : MY(6)=MY(0)
320    FOR J=0 TO 6
325       XO(J)=X(J):YO(J)=Y(J)
330       X(J)=MX(J) : Y(J)=MY(J)
340    NEXT J
350 NEXT N
360 A$=INPUT$(1)
370 COLOR 15,4,4 : END
```

Programma 19

We kunnen van de vliegekop, die programma 19 tekent, een 'echte' vliegekop met ogen maken. Hiertoe tekenen we twee ogen (cirkels) en kleuren het vlak dat door de cirkels en de daarbuitenliggende lijnen begrensd wordt. Het resultaat en de benodigde extra opdrachten in het programma staan hieronder.

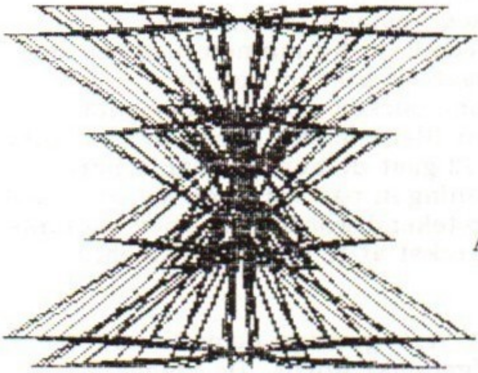


```
221 CIRCLE (FNX(148),96),10,1,0,2*PI,
                                  1.4
222 CIRCLE (FNX(108),96),10,1,0,2*PI,
                                  1.4
223 PAINT (FNX(163),100),1,1
224 PAINT (FNX(93),100),1,1
```

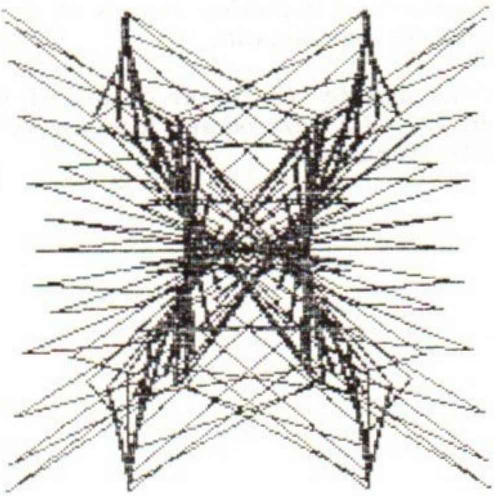
Neem gerust een andere kleur door in de PAINT-opdracht de 1 te vervangen door een getal tussen 0 en 15.

Programma 21

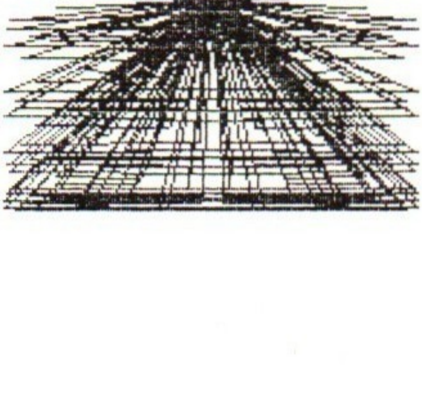
Allereerst geven we een paar voorbeelden van wat je zoal met programma 21 kunt tekenen. Je zou dit wellicht 'computerkunst' mogen noemen.



A=1/B=-100/C=-200



A=-300
B=-1
C=299



A=3/B=100/C=200

## Tekst op het grafische scherm

Het zou leuk zijn op het grafische scherm de waarden voor A, B en C af te drukken naast de tekening. Bevalt zo'n tekening, dan kunnen we direct de waarden van A, B en C noteren. In de regels 231-238 van het onderstaande programma zien we wat we hiervoor moeten doen. In regel 231 openen we een bestand (met nummer 1) voor uitvoer naar het grafische scherm (GRP:). We kunnen niet van de PRINT-opdracht gebruik maken, maar we moeten met de DRAW "BM..."-opdracht de grafische cursor naar een bepaalde schermpositie brengen (BM betekent Blank Move; alleen de grafische cursor verplaatst zich). In regel 232 gaat de cursor naar schermpositie (10,100). Aangezien de tekeningen pas vanaf de x-coördinaat 37 (zie FNX-functie in regel 170) getekend worden, hebben we links nog een veld van 0-36 bij 0-192 om tekst af te drukken. Rechts hebben we ook nog zo'n veld!

Regel 233 zorgt ervoor dat de tekst 'A: ' gevolgd door de waarde voor A (hoogstens vier posities) afgedrukt wordt. De volgende regels doen hetzelfde voor B en C. In regel 238 sluiten we het tekstuitvoerbestand.

Denk aan de '#1' achter de PRINT-opdrachten in de regels 233, 235 en 237. Via deze omweg kunnen we dus toch op een grafisch scherm iets afdrukken.

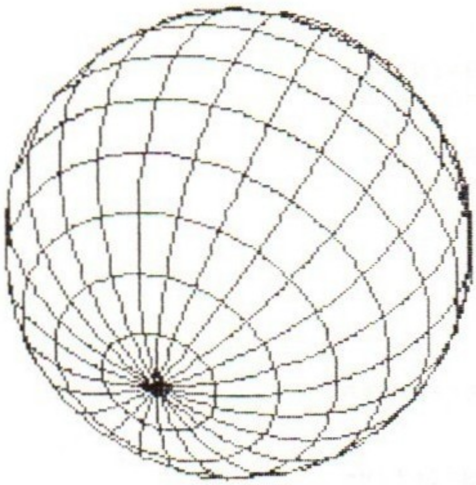
```
100 REM programma 21 symmetrische
                     figuren
110 CLS
120 PI=4*ATN(1)
130 U=128:V=92:H=.5:RD=PI/180:K=100
140 INPUT "toets A,B en C in"; A,B,C
150 CLS
160 COLOR 1,15,4 : SCREEN 2
170 DEF FNX(X)=INT(37+X/1.4+.5)
180 FOR W=0 TO 360
190   T=W*RD:R=K*SIN(C*T)
200   X2=INT(U+R*COS(A*T)+H)
210   Y2=INT(V-R*SIN(B*T)+H)
220    IF W=0
         THEN X1=X2:Y1=Y2
         ELSE LINE(FNX(X1),Y1)-
                  (FNX(X2),Y2):
              X1=X2:Y1=Y2
230 NEXT W
231 OPEN "GRP:" AS #1
232 DRAW "BM10,100"
233 PRINT#1, USING "A: ####";A
234 DRAW "BM10,108"
235 PRINT#1, USING "B: ####";B
236 DRAW "BM10,116"
237 PRINT#1, USING "C: ####";C
238 CLOSE#1
240 A$=INPUT$(1)
250 COLOR 15,4,4 : END
```

## Programma 25

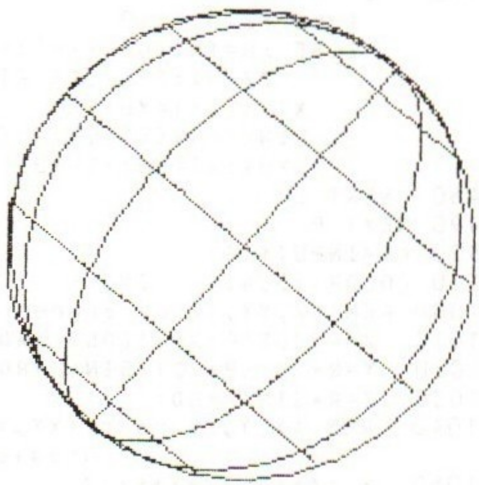
De auteur Marcel Sutter heeft onlangs in het tijdschrift Mikro+Kleincomputer een verbeterde versie van het 'bolprogramma' 25 gepubliceerd. Wij laten dit nu zien. Het programma tekent bollen in allerlei standen en met verborgen lijnen. De draaihoeken om de x-, y- en z-as alsmede de 'afstand' tussen de breedte- en lengtelijnen kunnen worden ingevoerd. Eerst volgt het programma, daarna enkele voorbeelden.

```
100 REM bol met hidden lines en
        draaiing
110 CLS
120 PRINT "bol met lengte en breedte"
        ;" lijnen"
130 PRINT:PRINT
140 INPUT "draaihoek om x-as";A
150 INPUT "draaihoek om y-as";B
160 INPUT "draaihoek om z-as";C
170 INPUT "afstand lijnen(10-45)";D
180 PI=4*ATN(1)
190 U=128:V=96:R=90:RD=PI/180:H=.5
200 S1=SIN(A*RD):S2=SIN(B*RD):
    S3=SIN(C*RD)
210 C1=COS(A*RD):C2=COS(B*RD):
    C3=COS(C*RD)
220 '
230 REM rotatiematrix berekenen
240 AX=C2*C3:AY=-C2*S3:AZ=S2
250 BX=C1*S3+S1*S2*C3
260 BY=C1*C3-S1*S2*S3:BZ=-S1*C2
270 CX=S1*S3-C1*S2*C3
280 CY=S1*C3+C1*S2*S3:CZ=C1*C2
290 REM hoge resolutie inschakelen
295 CLS
300 COLOR 1,15,4 : SCREEN 2
310 DEF FNX(X)=INT(37+X/1.4+.5)
320 REM bolomtrek tekenen
330 CIRCLE(FNX(U),V),R,1,0,2*PI,1.4
340 REM lengtelijnen tekenen
350 FOR L=0 TO 180-D STEP D
360  F1=0
370  FOR P=0 TO 360 STEP 5
380   GOSUB 1000 'XX,YY,ZZ berekenen
```

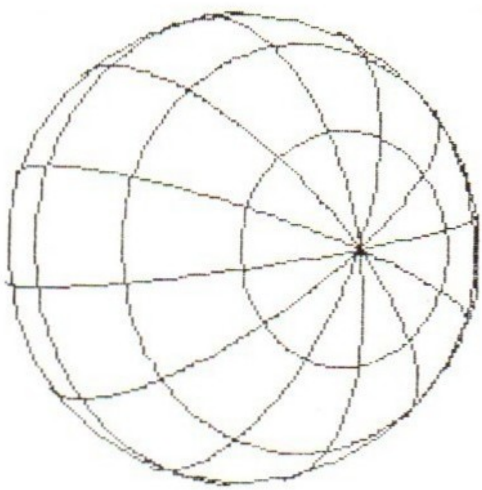
```
390    IF YY>0
        THEN F2=0:F1=0
        ELSE XB=INT(U+XX+H):YB=INT(V-
             ZZ+H):F2=1:IF F1=0 THEN
          X1=XB:Y1=YB:F1=1      ELSE
          LINE(FNX(X1),Y1)-(FNX(XB)
          ,YB):X1=XB:Y1=YB:F1=F2
400   NEXT P
410  NEXT L
420  REM breedtelijnen tekenen
430  FOR P=-90+D TO 90-D STEP D
440   F1=0
450   FOR L=0 TO 360 STEP 5
460    GOSUB 1000 'XX,YY,ZZ berekenen
470    IF YY>0
        THEN F2=0:F1=0
        ELSE XB=INT(U+XX+H):YB=INT(V-
             ZZ+H):F2=1:IF F1=0 THEN
          X1=XB:Y1=YB:F1=1      ELSE
          LINE(FNX(X1),Y1)-(FNX(XB)
          ,YB):X1=XB:Y1=YB:F1=F2
480   NEXT L
490  NEXT P
500  A$=INPUT$(1)
510  COLOR 15,4,4 : END
1000  REM XX,YY,ZZ berekenen
1010   X=R*COS(P*RD)*COS(L*RD)
1020   Y=R*COS(P*RD)*SIN(L*RD)
1030   Z=R*SIN(P*RD)
1040   REM (X,Y,Z) naar (XX,YY,ZZ)
                          draaien
1050   XX=AX*X+AY*Y+AZ*Z
1060   YY=BX*X+BY*Y+BZ*Z
1070   ZZ=CX*X+CY*Y+CZ*Z
1080  RETURN
```



draaihoek om x-as: -50
draaihoek om y-as: 20
draaihoek om z-as: 60
afstand lijnen : 15



draaihoek om x-as: 0
draaihoek om y-as: 40
draaihoek om z-as: 60
afstand lijnen : 30



draaihoek om x-as: 90
draaihoek om y-as: 30
draaihoek om z-as: 40
afstand lijnen : 30

**Programma 36**

We gaan de kaart van Zwitserland wat verfraaien. Eerst geven we het gewijzigde programma.

```
100 REM programma 36 gekleurde kaart
        van zwitserland met vlag
110 CLS
120 COLOR 6,15,15 : SCREEN 2
130 DEF FNX(X)=INT(37+X/1.4+.5)
140 K=1.35:H=.5:U=18:V=191
150 READ X,Y
160 X1=INT(U+K*X+H)
170 Y1=INT(V-K*Y+H)
180 REM begin herhaal lus
190     X2=INT(U+K*X+H)
200     Y2=INT(V-K*Y+H)
210     LINE (FNX(X1),Y1)-(FNX(X2),Y2)
220     X1=X2:Y1=Y2
230     READ X,Y
240     IF X<> 0 THEN 190
250 REM einde herhaal lus
260 A$=INPUT$(1)
261 IF A$="p" THEN PAINT(FNX(128),96)
                ,6,6:A$=INPUT$(1)
262 IF A$="v" THEN U=128:V=96:K=1:
        COLOR15,11,11:    GOTO 150
263 IF A$="k" THEN PAINT(FNX(128),96)
                ,15,15:A$=INPUT$(1)
270 COLOR 15,4,4 : END
280 '
290 DATA ...........................
     .
     .
520 DATA 0,0
530 DATA -40,10,-10,10,-10,40,10,40
540 DATA 10,10,40,10,40,-10,10,-10
550 DATA 10,-40,-10,-40,-10,-10,-40
560 DATA -10,-40,10,0,0
```

Wat zijn nu de veranderingen?

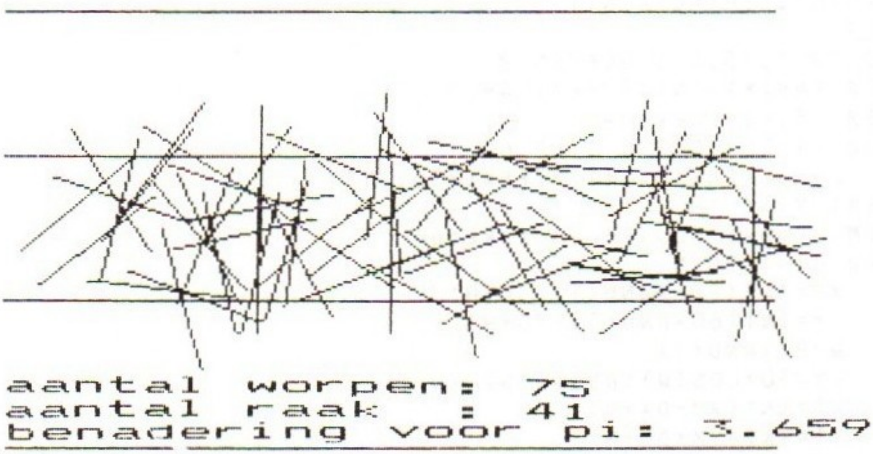
| | |
|---|---|
| regel 120: | COLOR 6,15,15; afdrukkleur rood, achtergrond en kader wit |
| regel 261: | IF A$="p" THEN PAINT(FNX(128),96),6,6; als we nu een kleine p intoetsen, wordt Zwitserland rood gekleurd |
| regel 262: | IF A$="v" THEN U=128:V=96:K=1: COLOR 15,11,11:GOTO 150; als we nu een kleine v intoetsen, wordt een wit omrand kruis (de vlag van Zwitserland) op het rode vlak getekend en het kader boven en onder de tekening wordt groen. Door GOTO 150 worden de extra data-regels 530-560 gelezen. Dit zijn de coördinaten van de hoekpunten van het kruis. |
| regel 263: | IF A$="k" THEN PAINT(FNX(128,96),15,15; als we nu een kleine k intoetsen, wordt het kruis wit gemaakt |

Het effect is nu als op de onderstaande tekening: wat daar zwart is, is op het scherm wit; wat wit is, is op het scherm rood en boven en onder de tekening zien we op het scherm nog twee stroken groen.



Programma 39

Dit educatieve programma vraagt om het afdrukken van de benaderde waarde van pi op het grafische scherm, onder de tekening met de 'gevallen' spelden. We willen eigenlijk dit zien:




De regels 361-369 zorgen ervoor dat de tekst en de waarden uit de PRINT-opdrachten in de regels 400, 410 en 420 van het programma op p.106 nu op het grafische scherm worden afgedrukt.

```
100 REM programma 39 speldenworp van
                    buffon
110 CLS
120 PRINT "speldenworp van Buffon"
130 PRINT "----------------------"
140 PRINT:PRINT
150 INPUT "hoeveel worpen";N
160 M=0:H=.5:PI=4*ATN(1)
170 CLS
180 COLOR 1,15,4 : SCREEN 2
190 DEF FNX(X)=INT(37+X/1.4+.5)
200 REM lijnen tekenen
210 FOR Y1=10 TO 190 STEP 60
220    LINE (FNX(0),Y1)-(FNX(256),Y1)
230 NEXT Y1
240 REM n maal gooien en tekenen
250 FOR J =1 TO N
260    XM=INT(226*RND(1)+30+H)
270    YM=INT(60*RND(1)+70+H)
280    W=PI*RND(1)
290    DX=30*COS(W):DY=30*SIN(W)
300    X1=INT(XM-DX+H)
310    Y1=INT(YM+DY+H)
320    X2=INT(XM+DX+H)
330    Y2=INT(YM-DY+H)
340    LINE (FNX(X1),Y1)-(FNX(X2),Y2)
350    IF Y1>130 OR Y2<=70
         THEN M=M+1
360 NEXT J
361 OPEN "grp:" AS #1
362 DRAW "BM37,160"
363 PRINT#1,"aantal worpen:";N
364 DRAW "BM37,168"
365 PRINT#1,"aantal raak  :";M
366 DRAW "BM37,176"
367 PRINT#1,"benadering voor pi:";
368 PRINT#1, USING " #.###";2*N/M
369 CLOSE#1
```

Het zal duidelijk zijn dat de in dit boek gegeven programma's onbeperkt uitgebreid kunnen worden. Ze kunnen vast ook verbeterd worden; de invoerwaarden zouden bijvoorbeeld gecontroleerd kunnen worden, zodat het programma niet ergens halverwege afbreekt door een foutieve schermcoördinaat. Wij hebben dat allemaal niet gedaan, omdat we de programma's klein wilden houden, waardoor alle aandacht op de tekentechniek gericht blijft.

## ACADEMIC SERVICE INFORMATICA UITGAVEN

### AUTOMATISERING EN COMPUTERS

*Computers en onze samenleving* - M.A. Arbib
*Computers in de negentiger jaren* - G.L. Simons
*De informatiemaatschappij* - Jan Everink
*Basiskennis informatieverwerking* - Jan Everink
*AIV, Automatisering van de informatieverzorging* - Th.J.G. Derksen en H.W. Crins
*Organisatie, informatie en computers* - D.M. Kroenke
*De Viewdata revolutie* - S. Fedida en R. Malik

### MICROCOMPUTERS

*Microcomputers thuis en op school* - K.P. Goldberg en R.D. Sherwood
*Bouw zelf een Expertsysteem in BASIC* - C. Naylor
*Programmeercursus Microsoft BASIC* - Nok van Veen
*Werken met bestanden in BASIC* - L. Finkel en J.R. Brown
*40 Grafische programma's voor de Commodore 64* - M. Sutter
*Doe-het-zelf programma's op de Commodore 64* - D. Kreutner
*Programmeercursus BASIC op de Commodore 64* - Nok van Veen
*TRS-80 BASIC* - Bob Albrecht e.a.
*TRS-80 BASIC voor gevorderden* - Don Inman e.a.
*Exidy sorcerer en BASIC* - Nok van Veen e.a.
*40 Grafische programma's voor de Electron en BBC* - M. Sutter
*Het Electron en BBC Micro boek* - Jim McGregor en Alan Watt
*Ontdek de ZX-Spectrum* - Tim Hartnell
*Werken met bestanden op de Apple* - L. Finkel en J.R. Brown
*Programmeercursus Applesoft BASIC* - Nok van Veen
*40 Grafische programma's voor de Apple II,IIe,IIc* - M. Sutter
*40 Grafische programma's in MSX BASIC* - M. Sutter
*Programmeercursus MSX BASIC* - Nok van Veen

### MICROPROCESSORS EN ASSEMBLEERTALEN

*Procescomputers, basisbegrippen* - dr.ir. J.E. Rooda en ir. W.C. Boot
*Cursus Z-80 assembleertaal* - Roger Hutty
*6502 Assembleertaal en machinecode voor beginners* - A.P. Stephenson

### BESTURINGSSYSTEMEN

*Inleiding besturingssystemen* - A.M. Lister
*Systeemprogrammatuur en software-ontwikkeling voor microcomputers* - E. Verhulst
*Bedrijfssystemen* - EIT-serie, deel 4
*CP/M het operating system voor microcomputers* - J.N. Fernandez en R. Ashley
*CP/M 86* - Nok van Veen
*CP/M voor gevorderden* - A. Clarke e.a.
*PC DOS, het besturingssysteem van de IBM PC* - R. Ashley en J.N. Fernandez
*MS/DOS, het besturingssysteem voor 16 bit microcomputers* - R. Ashley en J.N. Fernandez
*UNIX, het standaard operating system* - G.J.M. Austen en H.J. Thomassen
*Werken met UNIX* - Brian W. Kernighan en Rob Pike

### PERSONAL COMPUTERS

*Het werken met bestanden op de IBM PC* - L. Finkel en J.R. Brown
*De IBM PC en zijn toepassingen* - Laurence Press
*40 Grafische programma's voor de IBM PC* - M. Sutter
*Werken met VisiCalc* - C. Klitzner en M.J. Plociak
*Multiplan, een hulpmiddel bij de bedrijfsvoering* - D.F. Cobb e.a.
*Multiplan diskettes*
*Werken met Lotus 1-2-3* - D. Cobb en G. LeBlond

### PROGRAMMEREN

*Een methode van programmeren* - prof.dr. Edsger W. Dijkstra en ir. W.H.J. Feijen
*Programmeren, het ontwerpen van algoritmen (met Pascal)* - ir. J.J. van Amstel
*Inleiding tot het programmeren, deel 1* - ir. J.J. van Amstel
*Inleiding tot het programmeren, deel 2* - ir. J.J. van Amstel
*Programmeren, deel 2: van analyse tot algoritme* - prof.drs. C. Bron
*Inleiding programmeren en programmeertechnieken* - EIT-serie, deel 1
*Het Groot Pascal Spreuken Boek* - H.F. Ledgard e.a.
*JSP - Jackson Struktureel Programmeren* - Henk Jansen
*JSP Uitwerkingenboek* - Henk Jansen

## PROGRAMMEERTALEN

*Aspecten van programmeertalen* - ir. J.J. van Amstel en ir. J.A.A.M. Poirters
*Programmeertalen, een inleiding* - ir. J.J. van Amstel e.a.
*BASIC* - EIT-serie, deel 3
*Cursus BASIC, een practicum-handleiding voor BASIC op de PRIME* - ir. R. Bloothoofd e.a.
*Cursus Pascal* - prof.dr. A. van der Sluis en drs. C.A.C. Görts
*Cursus eenvoudig Pascal* - prof.dr. A. van der Sluis en drs. C.A.C. Görts
*Inleiding programmeren in Pascal* - C. van de Wijgaart
*Systeemontwikkeling met Ada* - Grady Booch
*Cursus COBOL* - A. Parkin
*Cursus FORTRAN 77* - J.N.P. Hume en R.C. Holt
*Aanvulling cursus FORTRAN 77 voor PRIME-computers* - ing. J.M. den Haan
*De programmeertaal C* - ir. L. Ammeraal
*Flitsend Forth* - Alan Winfield
*Programmeren in LISP* - prof.dr. L.L. Steels

## GEGEVENSSTRUCTUREN EN BESTANDSORGANISATIE

*Informatiestructuren, bestandsorganisatie en bestandsontwerp* - EIT-serie, deel 5
*Programmeren, het ontwerpen van datastructuren en algoritmen* - ir. J.J. van Amstel e.a.
*Bestandsorganisatie* - prof.dr. R.J. Lunbeck en drs. F. Remmen

## DATABASE EN GEGEVENSANALYSE

*Database, een inleiding* - C.J. Date
*Databases* - drs. F. Remmen
*Gegevensanalyse* - R.P. Langerhorst

## INFORMATIE-ANALYSE EN SYSTEEMONTWERP

*Effectieve toepassingen van computers* - M. Peltu
*Voorbereiding van computertoepassingen* - prof.dr. A.B. Frielink
*Systeemontwikkeling volgens SDM* - H.B. Eilers
*Samenvatting SDM* - Pandata
*Informatie-analyse volgens NIAM* - J.J.V.R. Wintraecken
*Evaluation of methods and techniques for the analysis, design and implementation of information systems* - ed. J. Blank en M.J. Krijger
*Inleiding systeemanalyse, systeemontwerp* - W.S. Davis
*Systeemontwikkeling Zonder Zorgen* - Paul T. Ward
*Het ontwerpen van interactieve toepassingen en computernetwerken* - J.A. Scheltens
*EDP Audit* - prof.dr. C. de Backer
*Prototyping, een instrument voor systeemontwerpers* - ed. T. Hoenderkamp en H.G. Sol
*Simulatie, een moderne methode van onderzoek* - drs. S.K. Boersma en ir. T. Hoenderkamp

## EXPERT SYSTEMEN EN KUNSTMATIGE INTELLIGENTIE

*Computerschaak* - H.J. van den Herik
*Expert systemen* - Henk de Swaan Arons en Peter van Lith

## THEORETISCHE INFORMATICA EN SYSTEEMPROGRAMMATUUR

*Informatica, een theoretische inleiding* - dr. L.P.J. Groenewegen en prof.dr. A. Ollongren
*Systeemprogrammatuur* - drs. H. Alblas
*Vertalerbouw* - H. Alblas e.a.

## AANVERWANTE ONDERWERPEN EN OVERIGE TITELS

*Lineaire programmering als hulpmiddel bij de besluitvorming* - prof.dr. S.W. Douma
*Inleiding programmeren* - prof.dr. R.J. Lunbeck
*Analyse van informatiebehoeften en de inhoudsbeschrijving van een databank* - prof.dr. P.G. Bosch en ir. H.M. Heemskerk
*Gegevensstructuren* - R. Engmann e.a.
*Cases en Uitwerkingenboek bij Cases* - prof.dr. P.G. Bosch en H.A. te Rijdt
*De tekstmachine* - dr. M. Boot en drs. H. Koppelaar
*Abstracte automaten en grammatica's* - prof.dr. A. Ollongren en ir. Th.P. van der Weide
*Onderneming en overheid in systeem-dynamisch perspectief* - red. A.F.G. Hanken e.a.
*Simulatie en sociale systemen* - red. J.L.A. Geurts en J.H.L. Oud
*Struktuur en stijl in COBOL* - ir. E. Dürr en dr.ir. F. Mulder
*Cursus ALGOL 60* - prof.dr. A. van der Sluis en drs. C.A.C. Görts

## Over de inhoud van dit boek

Dit boek bevat 40 grafische programma's voor het programmeren met hoge resolutie in MSX BASIC. Dit betekent dat het scherm van uw MSX-computer verdeeld wordt in 256 puntjes horizontaal en 192 puntjes verticaal. Op dit scherm kunnen fraaie tekeningen gemaakt worden.

Bij bijna elk programma wordt de nodige tekst en uitleg gegeven. Daarnaast bevat het boek veel illustraties van datgene wat de programma's op het scherm tekenen. Alle programma's zijn getest op een MSX-computer en draaien foutloos. De programmatekst is direct vanuit het geheugen van de computer op de printer afgedrukt en zo in het boek opgenomen.

De fraaie figuren berusten dikwijls op enige wiskundige vergelijkingen. In het boek wordt hier ook aandacht aan besteed. Dit vormt voor de niet-wiskundige lezer echter geenszins een belemmering het boek te gebruiken. Het alleen maar intikken en draaien van de programma's geeft al zulke fraaie beelden dat het bestuderen van de erachterliggende theorie geenszins noodzakelijk is.

Naast het tekenen van allerlei eenvoudige en meer ingewikkelde functies (scholieren en leraren zullen nu hun oren spitsen) worden de beginselen van het driedimensionaal tekenen uitgelegd. Vijf programma's laten zien wat voor figuren je met de taal LOGO kunt tekenen. Hieronder zijn ook de beroemde turtlegraphics. Ook bevat het boek vijf educatieve toepassingsprogramma's, waarin graphics de hoofdrol spelen.

In een appendix wordt een aantal programma's uitgebreid om de lezer te laten zien dat de mogelijkheden met MSX-graphics haast onbegrensd zijn. Er zijn voorbeelden van het werken met kleuren (COLOR en PAINT) en van het afdrukken van tekst en getallen op het grafische scherm.

**De illustratie op de voorkant is gemaakt met het onderstaande programma.**

```
100 REM programma 32 turtle-grafiek
                    (LOGO-2)
110 INPUT "coordinaten start ";X1,Y1
120 INPUT "beginrichting(gr) ";W
130 INPUT "verplaatsing      ";S
140 INPUT "draaihoek linksom ";DW
150 INPUT "toename zijde     ";DS
160 PI=4*ATN(1)
170 H=.5:RD=PI/180:W1=W*RD
180 CLS
190 COLOR 1,15,4 : SCREEN 2
200 DEF FNX(X)=INT(37+X/1.4+.5)
210  X2=INT(X1+S*COS(W1)+H)
220  Y2=INT(Y1-S*SIN(W1)+H)
230  IF X2<0 OR X2>256 OR
       Y2<0 OR Y2>192
        THEN A$=INPUT$(1):STOP
240  LINE (FNX(X1),Y1)-(FNX(X2),Y2)
250  X1=X2:Y1=Y2
260  W=W+DW
270  IF W>360 THEN W=W-360
280  W1=W*RD:S=S+DS
290 GOTO 210
300 A$=INPUT$(1)
310 COLOR 15,4,4: END
```